**PESQUISA SOBRE BRANCH**

Primeiro, digamos que você esteja trabalhando no seu projeto e já tem alguns commits. Você decidiu que irá trabalhar na tarefa (issue) #53 do gerenciador de bugs ou tarefas que sua empresa usa. Para deixar claro, Git não é amarrado a nenhum gerenciador de tarefas em particular; mas já que a tarefa #53 tem um foco diferente, você criará um branch novo para trabalhar nele. Para criar um branch e mudar para ele ao mesmo tempo, você pode executar o comando git checkout com a opção -b:

$ git checkout -b iss53

Switched to a new branch "iss53"

Isso é um atalho para:

$ git branch iss53

$ git checkout iss53

Você trabalha no seu web site e faz alguns commits. Ao fazer isso o branch iss53 avançará, pois você fez o checkout dele (isto é, seu HEAD está apontando para ele;).

$ vim index.html

$ git commit -a -m 'adicionei um novo rodapé [issue 53]'

O branch iss53 avançou com suas modificações.

Nesse momento você recebe uma ligação dizendo que existe um problema com o web site e você deve resolvê-lo imediatamente. Com Git, você não precisa fazer o deploy de sua correção junto com as modificações que você fez no iss53, e você não precisa se esforçar muito para reverter essas modificações antes que você possa aplicar sua correção em produção. Tudo que você tem a fazer é voltar ao seu branch master.

No entanto, antes de fazer isso, note que seu diretório de trabalho ou área de seleção tem modificações que não entraram em commits e que estão gerando conflitos com o branch que você está fazendo o checkout, Git não deixará você mudar de branch. É melhor ter uma área de trabalho limpa quando mudar de branch. Existem maneiras de contornar esta situação (isto é, incluir e fazer o commit) que vamos falar depois. Por enquanto, você fez o commit de todas as suas modificações, então você pode mudar para o seu branch master:

$ git checkout master

Switched to branch "master"

Nesse ponto, o diretório do seu projeto está exatamente do jeito que estava antes de você começar a trabalhar na tarefa #53, e você se concentra na correção do erro. É importante lembrar desse ponto: Git restabelece seu diretório de trabalho para ficar igual ao snapshot do commit que o branch que você criou aponta. Ele adiciona, remove, e modifica arquivos automaticamente para garantir que sua cópia é o que o branch parecia no seu último commit nele.

Em seguida, você tem uma correção para fazer. Vamos criar um branch para a correção (hotfix) para trabalhar até a conclusão.

$ git checkout -b 'hotfix'

Switched to a new branch "hotfix"

$ vim index.html

$ git commit -a -m 'consertei o endereço de email'

[hotfix]: created 3a0874c: "consertei o endereço de email"

1 files changed, 0 insertions(+), 1 deletions(-)

Você pode rodar seus testes, tenha certeza que a correção é o que você quer, e faça o merge no seu branch master para fazer o deploy em produção. Você faz isso com o comando git merge:

$ git checkout master

$ git merge hotfix

Updating f42c576..3a0874c

Fast forward

README | 1 -

1 files changed, 0 insertions(+), 1 deletions(-)

Você irá notar a frase "Fast forward" no merge. Em razão do branch que você fez o merge apontar para o commit que está diretamente acima do commit que você se encontra, Git avança o ponteiro adiante. Em outras palavras, quando você tenta fazer o merge de um commit com outro que pode ser alcançado seguindo o histórico do primeiro, Git simplifica as coisas movendo o ponteiro adiante porque não existe modificações divergente para fazer o merge — isso é chamado de "fast forward".

Sua modificação está agora no snapshot do commit apontado pelo branch master, e você pode fazer o deploy.

Depois que a sua correção super importante foi enviada, você está pronto para voltar ao trabalho que estava fazendo antes de ser interrompido. No entanto, primeiro você apagará o branch hotfix, pois você não precisa mais dele — o branch master aponta para o mesmo local. Você pode excluí-lo com a opção -d em git branch:

$ git branch -d hotfix

Deleted branch hotfix (3a0874c).

Agora você pode voltar para o trabalho incompleto no branch da tafera #53 e continuar a trabalhar nele.

$ git checkout iss53

Switched to branch "iss53"

$ vim index.html

$ git commit -a -m 'novo rodapé terminado [issue 53]'

[iss53]: created ad82d7a: "novo rodapé terminado [issue 53]"

1 files changed, 1 insertions(+), 0 deletions(-)

Vale a pena lembrar aqui que o trabalho feito no seu branch hotfix não existe nos arquivos do seu branch iss53. Se você precisa incluí-lo, você pode fazer o merge do seu branch master no seu branch iss53 executando o comando git merge master, ou você pode esperar para integrar essas mudanças até você decidir fazer o pull do branch iss53 no master mais tarde.