Suponhamos que alguém da nossa equipe sobrescrevesse uma alteração que realizamos, modificando o título de "Lista de Cursos" para "Lista de Cursos de DevOps", inclusive commitando a alteração, e que, ainda, uma terceira pessoa da equipe, sabendo que esta alteração não deveria ter sido feita, modificasse o texto para "Lista de Cursos da Alura", commitando em seguida. Além disso, outra pessoa teria alterado o título depois de tudo isso, para "Cursos da Alura", e mais alguém voltaria o texto para "Lista de Cursos".

Muitas modificações foram feitas no título, e agora queremos encontrar o momento exato que determinada alteração foi aplicada. Isto quer dizer que queremos voltar ao estado em que o título esteve como "Lista de Cursos da Alura", por exemplo. Executando git log --oneline, notaremos que as mensagens de commits não são muito significativas e não nos ajudam nisso.

Seria necessário indicarmos ao Git que ele terá que buscar, dentre determinados commits — como os cinco últimos, por exemplo —, e ele teria que passar por cada um deles. Esta feature é chamada de **bisect**, e para executá-la digitaremos git bisect start, o que inicializará a busca do lado do Git. Inicialmente, é preciso informar a ele um estado, ou commit cuja parte do código que queremos não esteja boa, no caso, o título deverá ser "Lista de Cursos da Alura".

Vamos executar git bisect bad HEAD, e em seguida devemos informar o estado em que possivelmente estava bom, isto é, a partir de onde ele irá buscar o commit desejado. Então, copiaremos o hash referente à primeira alteração de título, e digitaremos git bisect good c17076a. Estes serão os limites da busca feita pelo Git. Obteremos a mensagem de que há uma revisão a mais para testarmos depois desta, e ele nos mostrará o estado de "Mudando título".

Neste estado, o título será aquele que queremos, e se ele ainda não tivesse encontrado a alteração desejada, continuaríamos executando git bisect bad, mas no nosso caso, como já estaremos no estado desejado, usaremos git bisect good. Após esta revisão, se tudo estiver bem, não precisaremos mais fazer nada. Para finalizar a busca, uma vez que o Git já nos entregou o hash e a descrição do commit, usaremos git bisect reset, com o qual voltaremos à master.

Com o hash do commit, poderíamos desfazer a alteração, analisar o porquê da inclusão dela, perguntar o que aconteceu à pessoa que realizou o commit. Para verificarmos todas as alterações referentes ao commit, aplicaremos o comando git show seguido do hash. Para reverter esta ação, pode-se utilizar git revert juntamente ao hash.

Notem como é relativamente fácil encontrar um commit em que uma alteração específica foi implementada, sem que precisássemos, por exemplo, usar o git checkout ou git show em cada um dos commits existentes. E se, a partir do momento em que encontramos uma alteração, ou quando ainda estivermos analisando o log de commits, quisermos saber quem foi o responsável por determinado commit, será que o GitHub consegue nos ajudar?

git blame mostra quem foi que fez as alterações, me da uma lista.

1. Execute o comando git bisect start para informar ao Git que você vai iniciar uma busca por determinada alteração;
2. Execute o comando git bisect bad HEAD para informar que o estado atual do código está "ruim", ou seja, o título não está no estado que você quer;
3. Executando git log --oneline, copie o *hash* do *commit* com a mensagem "Simplificando o título";
4. Execute o comando git bisect good {hash}, substituindo {hash} pelo *hash* copiado no passo anterior, para informar que o estado atual do código está "bom", ou seja, o título está no estado que você quer;
5. Confira no código que agora o título está como você quer;
6. Execute git bisect good para informar ao Git que neste *commit* o código ainda está como você quer;
7. Confira no código que agora o título não está mais como você quer;
8. Execute git bisect bad para informar ao Git que neste *commit* o código não está mais como você quer;
9. Note que o Git encontrou o exato *commit* onde o título deixou de estar no estado em que você quer;
10. Execute git bisect reset para indicar que você finalizou a busca;
11. Execute o comando git show {hash}, substituindo {hash} pelo *hash* conferido no passo 30. Veja que é exatamente a alteração que você estava buscando;
12. Para reverter esta alteração, execute git revert {hash}, substituindo {hash} pelo *hash* conferido no passo 30. Resolva quaisquer conflitos que possam ter sido encontrados;
13. Execute git blame index.html e confira o responsável por cada linha do arquivo.

Nesta aula, aprendemos:

* Que o comando git cherry-pick pode trazer um *commit* específico para a *branch* atual;
* Como encontrar o *commit* em que determinada alteração foi aplicada, utilizando o git bisect;
* Como encontrar o responsável por determinada linha ou bloco de código, utilizando o git blame;
* Que jamais devemos apontar um culpado por determinado *bug*. Uma equipe deve ser unida e se ajudar;
* Que o comando git show {hash} mostra todas as alterações aplicadas pelo *commit* com o *hash* informado.