Apostila de introdução à controle de versões e Git

Evaldo Junior Bento

April 10, 2011

Resumo

Pequenas e grandes equipes de desenvolvimento de *software* sofrem com problemas de versionamento de código fonte. Quem fez? O que fez? Quando fez? Quais linhas foram alteradas? Estas são perguntas comuns no dia a dia de equipes que não usam sistemas de controle de versão.

Esta apostila foi desenvolvida para ajudar iniciantes e interessados em sistemas de controle de versões à entender os conceitos de controle de versões e por que fazê-lo. O **Git**, desenvolvido por Linus Torvalds, criador do *kernel* Linux, é o software de controle de versões usado nesta apostila para exemplificar os conceitos.

Sumário

Capítulo 1

Introdução

Desenvolver *software* sem utilizar um sistema de controle de versões...

Capítulo 2

Fluxo comum de trabalho em um projeto de software

O fluxo "normal" de trabalho, em um projeto de *software*, pode ser resumido em:

- Pegar o código atual;
- Editar;
- Salvar;
- Devolver para o centralizador.

Até aqui, tudo bem. O desenvolvedor abre o código fonte, faz seu trabalho e então devolve o resultado para uma base central. Em projetos individuais ou de equipes bem pequenas, como duplas ou trios, esse método pode até funcionar, mesmo assim esta não é uma boa solução.

2.1 Os problemas começam

Imagine alterar um arquivo, colocar em produção e depois de um tempo ver que suas alterações simplesmente desapareceram. Isso pode acontecer quando outro desenvolvedor também fizer alterações no mesmo arquivo e enviar as suas alterações após o primeiro, sem antes verificar se os arquivos do projeto continuavam iguais aos que ele pegou antes de alterar.

Veja o tempo que se perde por ter que verificar os arquivos antes de cada atualização. E o tempo maior ainda por ter que refazer algo que já havia sido feito, fora a frustração e o desanimo que esse tipo de situação geralmente causa.

2.2 A solução aparece no horizonte

E se existisse algum tipo de software que ajudasse a verificar os arquivos, saber quem alterou, que linhas alterou e quando alterou?

Sim, isso seria muito legal. Mas espere, isso existe sim! São os Sistemas de Controle de Versão[1].

[1]SCM - Source Code Management, em inglês.

Capítulo 3

Controle de versões

O que é?

Capítulo 4

Git

Desenvolvido por Linus Torvalds

Capítulo 5

Iniciando um projeto

Para iniciar um novo projeto, crie um diretório, ou use um já existe, inclusive onde já existem arquivos, e use, neste diretório, a opção **init** do Git:

```
$ mkdir projeto
$ git init
```

Agora crie os arquivos ou trabalhe no seu projeto, normalmente. Até este momento o Git ainda não sabe da existência dos arquivos do projeto, para que os arquivos sejam adicionados ao controle de versões use a opção **add** do Git:

```
$ git add arquivo
```

Também é possível adicionar todos os arquivos novos/alterados de uma vez usando a opção **ponto**:

```
$ git add .
```

Depois de adicionar os arquivos é necessário fazer o *commit*[1]. Para fazer o commit use a opção **commit** do Git:

```
$ git commit –m "Mensagem descrevendo o que foi feito"
```

A opção **-m** usada no comando acima permite adicionar uma pequena mensagem de *commit* diretamente na linha de comando. Caso essa opção não seja informada será aberto o editor de textos padrão para que a mensagem seja digitada. Em geral o editor usado é o **Vim** ou o **Nano**. Optar por não usar o **-m** é uma boa escolha quando se deseja escrever mensagens de *commit* maiores e mais detalhadas.

[1]Commit: É uma confirmação do que foi feito.