**Sobre Controle de Versão**

O que é controle de versão, e por que você deve se importar? O controle de versão é um sistema que registra as mudanças feitas em um arquivo ou um conjunto de arquivos ao longo do tempo de forma que você possa recuperar versões específicas. Mesmo que os exemplos desse livro mostrem arquivos de código fonte sob controle de versão, você pode usá-lo com praticamente qualquer tipo de arquivo em um computador.

Se você é um designer gráfico ou um web designer e quer manter todas as versões de uma imagem ou layout (o que você certamente gostaria), usar um Sistema de Controle de Versão (Version Control System ou VCS) é uma decisão sábia. Ele permite reverter arquivos para um estado anterior, reverter um projeto inteiro para um estado anterior, comparar mudanças feitas ao decorrer do tempo, ver quem foi o último a modificar algo que pode estar causando problemas, quem introduziu um bug e quando, e muito mais. Usar um VCS normalmente significa que se você estragou algo ou perdeu arquivos, poderá facilmente reavê-los. Além disso, você pode controlar tudo sem maiores esforços.

### [Sistemas de Controle de Versão Locais](https://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o#Sistemas-de-Controle-de-Vers%C3%A3o-Locais)

O método de controle de versão de muitas pessoas é copiar os arquivos para outro diretório (talvez um diretório com carimbo de tempo, se eles forem espertos). Esta abordagem é muito comum, porque é tão simples, mas também é incrivelmente propenso a erros. É fácil esquecer em qual diretório você está e acidentalmente sobreescrever o arquivo errado ou copiar arquivos que não quer.

Para lidar com este problema, programadores há muito tempo desenvolveram VCSs locais que tem um banco de dados simples que mantêm todas as alterações nos arquivos sob controle de revisão.

Uma das ferramentas VCS mais populares foi um sistema chamado RCS, que ainda é distribuído com muitos computadores hoje. Até mesmo o popular sistema operacional Mac OS X inclui o comando rcsquando você instala as Ferramentas de Desenvolvimento. RCS funciona mantendo conjuntos de alterações (ou seja, as diferenças entre os arquivos) em um formato especial no disco; ele pode, em seguida, re-criar como qualquer arquivo se parecia em qualquer ponto no tempo, adicionando-se todas as alterações.

### Systemas Centralizados de Controle de Versão

A próxima questão importante que as pessoas encontram é que elas precisam colaborar com desenvolvedores em outros sistemas. Para lidar com este problema, Sistemas Centralizados de Controle de Versão (CVCSs) foram desenvolvidos. Estes sistemas, tais como CVS, Subversion e Perforce, têm um único servidor que contém todos os arquivos de controle de versão, e um número de clientes que usam arquivos a partir desse lugar central. Por muitos anos, este tem sido o padrão para controle de versão.

Esta configuração oferece muitas vantagens, especialmente sobre VCSs locais. Por exemplo, todo mundo sabe, até certo ponto o que todo mundo no projeto está fazendo. Os administradores têm controle refinado sobre quem pode fazer o que; e é muito mais fácil de administrar um CVCS do que lidar com bancos de dados locais em cada cliente.

No entanto, esta configuração também tem algumas desvantagens graves. O mais óbvio é o ponto único de falha que o servidor centralizado representa. Se esse servidor der problema por uma hora, durante essa hora ninguém pode colaborar ou salvar as alterações de versão para o que quer que eles estejam trabalhando. Se o disco rígido do banco de dados central for corrompido, e backups apropriados não foram mantidos, você perde absolutamente tudo - toda a história do projeto, exceto imagens pontuais que desenvolvedores possam ter em suas máquinas locais. Sistemas VCS locais sofrem com esse mesmo problema - sempre que você tenha toda a história do projeto em um único lugar, há o risco de perder tudo.

### Systemas Distribuídos de Controle de Versão

É aqui que Sistemas Distribuídos de Controle de Versão (DVCS) entram em cena. Em um DVCS (como Git, Mercurial, Bazaar ou Darcs), clientes não somente usam o estado mais recente dos arquivos: eles duplicam localmente repositório completo. Assim, se qualquer servidor morrer, e esses sistemas estiverem colaborando por meio dele, qualquer um dos repositórios de clientes podem ser copiado de volta para o servidor para restaurá-lo. Cada clone é de fato um backup completo de todos os dados.

Além disso, muitos desses sistemas trabalham muito bem com vários repositórios remotos, tal que você possa colaborar em diferentes grupos de pessoas de maneiras diferentes ao mesmo tempo dentro do mesmo projeto. Isso permite que você configure vários tipos de fluxos de trabalho que não são possíveis em sistemas centralizados, como modelos hierárquicos.