**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-Versão-Locais)

O método preferido de controle de versão por muitas pessoas é copiar arquivos em outro diretório (talvez um diretório com data e hora, se forem espertos). Esta abordagem é muito comum por ser tão simples, mas é também muito suscetível a erros. É fácil esquecer em qual diretório você está e gravar acidentalmente no arquivo errado ou sobrescrever arquivos sem querer. Para lidar com esse problema, alguns programadores desenvolveram há muito tempo VCSs locais que armazenavam todas as alterações dos arquivos sob controle de revisão

Uma das ferramentas de VCS mais populares foi um sistema chamado rcs, que ainda é distribuído em muitos computadores até hoje. Até o popular Mac OS X inclui o comando rcs quando se instala o kit de ferramentas para desenvolvedores. Basicamente, essa ferramenta mantém conjuntos de patches (ou seja, as diferenças entre os arquivos) entre cada mudança em um formato especial; a partir daí qualquer arquivo em qualquer ponto na linha do tempo pode ser recriado ao juntar-se todos os patches.

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

Outro grande problema que as pessoas encontram estava na necessidade de trabalhar em conjunto com outros desenvolvedores, que usam outros sistemas. Para lidar com isso, foram desenvolvidos Sistemas de Controle de Versão Centralizados (Centralized Version Control System ou CVCS). Esses sistemas, como por exemplo o CVS, Subversion e Perforce, possuem um único servidor central que contém todos os arquivos versionados e vários clientes que podem resgatar (check out) os arquivos do servidor. Por muitos anos, esse foi o modelo padrão para controle de versão.

Tal arranjo oferece muitas vantagens, especialmente sobre VCSs locais. Por exemplo, todo mundo pode ter conhecimento razoável sobre o que os outros desenvolvedores estão fazendo no projeto. Administradores têm controle específico sobre quem faz o quê; sem falar que é bem mais fácil administrar um CVCS do que lidar com bancos de dados locais em cada cliente.

Entretanto, esse arranjo também possui grandes desvantagens. O mais óbvio é que o servidor central é um ponto único de falha. Se o servidor ficar fora do ar por uma hora, ninguém pode trabalhar em conjunto ou salvar novas versões dos arquivos durante esse período. Se o disco do servidor do banco de dados for corrompido e não existir um backup adequado, perde-se tudo — todo o histórico de mudanças no projeto, exceto pelas únicas cópias que os desenvolvedores possuem em suas máquinas locais. VCSs locais também sofrem desse problema — sempre que se tem o histórico em um único local, corre-se o risco de perder tudo.

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

É aí que surgem os Sistemas de Controle de Versão Distribuídos (Distributed Version Control System ou DVCS). Em um DVCS (tais como Git, Mercurial, Bazaar ou Darcs), os clientes não apenas fazem cópias das últimas versões dos arquivos: eles são cópias completas do repositório. Assim, se um servidor falha, qualquer um dos repositórios dos clientes pode ser copiado de volta para o servidor para restaurá-lo. Cada checkout (resgate) é na prática um backup completo de todos os dados

Além disso, muitos desses sistemas lidam muito bem com o aspecto de ter vários repositórios remotos com os quais eles podem colaborar, permitindo que você trabalhe em conjunto com diferentes grupos de pessoas, de diversas maneiras, simultaneamente no mesmo projeto. Isso permite que você estabeleça diferentes tipos de workflow (fluxo de trabalho) que não são possíveis em sistemas centralizados, como por exemplo o uso de modelos hierárquicos.