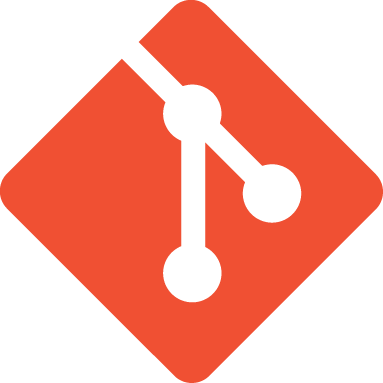
Git e GitHub

Git e GitHub são coisas, sistemas diferentes.

Git – VCS - Software de Controle de Versão



GitHub – Repositório de código e Rede Social para Programadores



Git

Versões de software – Versionamento de software

Qual a importância das versões de um software?

Organização – backup (segurança) – atualização – trabalho em equipe

Exemplo:

Ao iniciar o projeto do desenvolvimento de uma site para um cliente, o primeiro passo é: criar uma pasta e organizar os arquivos (html, css, javascript, assim como, imagens, vídeos entre outros) na mesma.

Após apresentar um primeiro esboço ao cliente, e o mesmo concordar com o rumo dado por você ao projeto, normalmente, o próximo passo é compactar a pasta do projeto. Inicialmente para fazer um backup, já que você está acertando no desenvolvimento.

Entretanto, o cliente sugere algumas modificações no site, e depois de realizar as alterações no código, você criar um novo arquivo compactado, agora chamado de versão 2.

E este processo pode se repetir pode diversas vezes durante o desenvolvimento do site para o cliente.

Pode ser que em algum momento, mesmo com um problema em alguma funcionalidade do site, você ainda decida por manter aquele estado, então compacta o projeto e o chama de versão com defeito.

E se em seguida você consegue resolver o problema, então criar outro arquivo compactado chamado de versão problema resolvido. Mas não apaga o arquivo da versão com defeito pois não sabe o que pode acontecer pela frente.

No final do projeto você cria um arquivo compactado versão final. Mas a versão final ainda pode sofrer alterações, e você acaba criando a versão final 1, versão final 2, e assim por diante.

Bom, depois de concluir o projeto você acaba ficando com um monte de versões daquele projeto, diversos arquivos compactados. E agora, o que fazer com estes backups? Provavelmente você pensa em guarda-los, pois os arquivos podem ser úteis em um novo projeto. Então, você copia os arquivos para um pendrive, ou grava em um CD/DVD, ou ainda deixa salvo no seu HD. Mas, e se você perder o pendrive? Ou se a sua mídia ficar arranhada? Ou se o seu HD estragar?

Como você não é bobo, a primeira coisa que você pensa é em salvar os arquivos no Google Drive, Dropbox, ou qualquer outra nuvem de armazenamento.

Passado algum tempo, o seu cliente te liga e solicita uma imagem que foi utilizada no início do desenvolvimento do site. E agora? O que fazer? Você terá que baixar todos os arquivos, descompacta-los, e localizar o arquivo com a imagem que o cliente está solicitando.

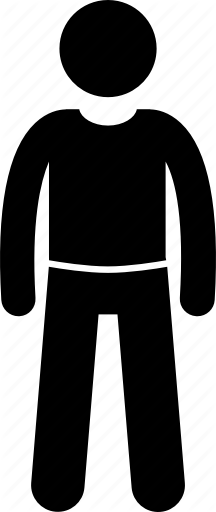
Agora, imagem outro cenário. Se você está desenvolvendo este projeto em equipe, com alguns colegas de aula. Neste cenário, cada pessoa fica responsável por uma parte do desenvolvimento: design, conteúdo, front-end, back-end..

Imagine agora que todos os envolvidos no projeto passam a salvar todos os seus backups na mesma nuvem. Imagine a quantidade de arquivos. Que confusão isso vai gerar!

Neste caso, e por mais simples que seja o projeto, o melhor a se fazer é utilizar um VCS (Controle de Versão de Software). Neste caso o Git, que é um software de versionamento de código.

O versionamento de software não é uma coisa nova. O versionamento de código surgiu em 1972. Então, esta é uma prática utilizada por programadores a bastante tempo.

Em um software de versionamento, a ação de enviar uma versão do software para um repositório central é denominada COMMIT.

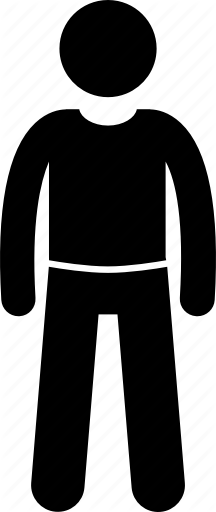
  

Repositório remoto

Nos primeiros softwares de versionamento, para que você pudesse realizar um COMMIT, o sistema precisaria de uma conexão ativa com o servidor constantemente. Deste modo, você não conseguiria manter o seu código monitorado se não tivesse uma conexão ativa com o servidor. E a conexão com o servidor não significa ter uma conexão com a Internet, principalmente na década de 70. Outras pessoas também podem participar do mesmo projeto, realizando o versionamento do código. E quem vai estar a cargo do versionamento será o servidor. Este tipo de VCS é chamado de Centralizado/Linear.

Este primeiro modelo evolui para o modelo distribuído.

No modelo distribuído, podemos ter a mesma equipe trabalhando no projeto, contudo, quando um dos desenvolvedores realizam um COMMIT este não será direcionado para o repositório central (o servidor), e sim para um repositório local. Neste modelo, você não precisa do poder de processamento de um servidor e muito menos estar conectado ao mesmo constantemente.

Repositório local repositório remoto

PUSH

Commit

Neste modelo, como as pessoas trabalham em seus repositórios locais e os demais desenvolvedores conseguem enxergar o que está sendo feito? Neste caso, além do repositório local, também será utilizado um repositório remoto.

Então, além de realizar um COMMIT para o seu repositório local a cada alteração que você realizar no projeto, após concluir uma tarefa ou finalizar o dia de trabalho, você também realizará um PUSH, que é o envio do versionamento local para o repositório remoto.

Exemplos de VCS

Centralizado/linear

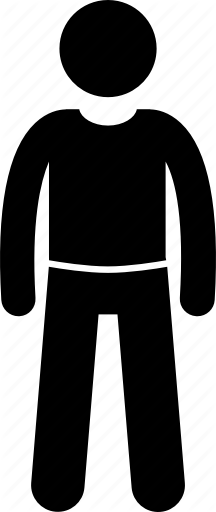
* CA Software Change Manager (Change and Configurantion Control)
* Source Code Control System (SCCS)
* Panvalet
* Concurrent Version System (CVS) – Um dos mais famosos
* Apache Subversion (SVN)
* ClearCase (IBM)
* Visual SourceSafe (Problemático – perda de dados)
* Perforce

Distribuído

* Mercurial
* Bazaar
* Code Co-op
* GNU arch
* Monotose
* Fossil
* BitKeeper
* Git

Principais vantagens

* Controle de histórico
* Trabalho em equipe
* Ramificação do projeto
* Segurança
* Organização

Repositório local repositório remoto

