Aula Exercicio de Introducao ao Git

Thiago de Gouveia Nunes

19 de outubro de 2011

1 Controle de Versão

Um controle de versão é um sistema responsável por gravar as mudanças em um ou mais arquivos. Temos dois tipos de software de controle de versão, os que usão o para digma centralizado e o distribuído. O SVN e o CVS são exemplos de sistemas centralizados. Eles guardão todos os dados de um projeto em um servidor central, online. O git e o mercurial são exemplos de sistemas distribuidos. Eles, além de usarem repositórios online, criam um repositório local no seu computador.

2 O que é o git

Enquanto a maioria dos SCV guardam as informações como um arquivo base e as suas modificações, o git guarda uma "foto"das suas modificações. Toda vez que você modifica ou salva algo no git, ele tira uma "foto"de todo o seu espaço de tra balho.

A maioria do trabalho do git é local. O git salva todas as modificações dos arquivos localmente, e toda vez que você baixar as atualizações do seu projeto, todo o histórico de modificações vem junto.

O git tem integridade. Isso quer dizer que é impossível modificar arquivos sem o git saber que você está fazendo algo. O git usa um mecanismo chamado SHA-1 hash. O git quase sempre adiciona dados. Na maioria das vezes, o git só adiciona dados no repositório. Assim, você pode sempre resgatar arquivos que já foram deletados a muito tempo.

O git usa uma mecânica de 3 estagios para controlar os arquivos na sua área de trabalho. Estes estagios sao: commited, modified e staged. Commited significa que o arquivo esta na sua database local. Modificado quer dizer que o arquivo foi modificado. Staged quer dizer que o arquivo foi selecionado para entrar no seu próximo commit.

3 Como iniciar um rep no git

O github tem um bom tutorial para instalar e iniciar o git:

- Windows http://help.github.com/win-set-up-git/ (Sem o tortoise git. Há um tutorial breve sobre o turtoise git abaixo.)
- Linux http://help.github.com/linux-set-up-git/
- MacOS http://help.github.com/mac-set-up-git/

4 Iniciar o git

A primeira coisa a se fazer depois de instalar o git e colocar o seu nome de usuario e e-mail, assim todo commit que voce fizer estara com essas informacoes. Para isso, use os comandos:

```
$git config --global user.name "Seu Nome"
$git config --global user.email seuemail@exemplo.comando
```

Para configurar a sua ferramenta de diff, use o comando:

```
$git config --global merge.tool NomeDaFerramenta
```

Obs: O git aceita kdiff3, tkdiff, meld, xxdiff, emerge, vimdiff, gvimdiff, ecmerge e opendiff como ferramentas de merge.

5 Help no git

Para acessar o manual do git (e de seus comandos) voce pode usar qualquer uma das sintaxes abaixo:

```
$git help <comando>
$git <comando> --help
$man git-<comando>
```

6 Como usar o git

Agora que o git está instalado e o repositório já está criado, podemos começar a modificar nosso projeto. O git usa um mecânismo para controlar as modificações nos seus arquivos, e usa um sistema um pouco diferente dos outros controles de versões para mandar essas modificações para o remoto.

6.1 Arquivos no Git

Todo arquivo, antes de ser adicionado no seu repositório, é reconhecido no git como Untracked. Ao ser adicionado usando o comando:
`$ git add <Nome-do-Arquivo>`
ele entra num ciclo de 3 fases do git.

Todo arquivo no git tem dois status: Tracked e Untracked. Arquivos Tracked serão adicionados no próximo commit. Para verifica o status de cada arquivos usamos o comando git status. O comando git commit adiciona os arquivos Tracked para o seu repositório. Toda vez que um arquivo é modificado ele é marcado como Untracked. Para mudar o status de um arquivo para Tracked usamos o comando git add `<Nome-do-Arquivo>`. O comando git diff mostra a diferença entre os arquivos Untracked e os Tracked. Um arquivo pode estar Tracked e Untracked ao mesmo tempo se dermos git add nele, modificarmos ele, e executar mos o git status, ele vai mostrar o mesmo arquivo nas duas partes, a que mostra os arquivos que serão commitados e os que não serão. Isso quer dizer que a versão não modificada do seu arquivo vai ser commitada, e não a mais nova. Para corrigir isso teriamos que dar git add no mesmo arquivo. Para remover um arquivo do git usamos o comando git rm ¡Nome-do-Arquivo¿. O comando `git mv file_from file_to` move um arquivo. O git commit só modifica os arquivos no seu repositorio local. Precisamos usar o comando git push para mandar as modificações para o repositorio no github. Se a sua versão for mais nova que a do repositorio, o seu push será feito e as modificações serão feitas. Se não, usamos o comando git pull para pegar a versão mais nova do repositorio e dar um merge com o nosso repositorio. Se houver algum problema com o merge, por exemplo um arquivo que você modificou foi modi ficado no repositorio no github, você terá que abrir o arquivo e modifica-lo manualmente para arrumá-lo. Depois disso, o git push poderá ser executado sem problema.

7 Branches

Branches sao facilmente criados no git. Eles sao arquivos de 41bits que apon tam para as mudancas feitas dentro deles. Vale lembrar que um branch criado por voce e local, e ele so e adicionado no repositorio local se usarmos o push desta maneira: git push (remote) (branch) exemplo: git push origin teste, isso vai fazer o branch teste ser integrado no nosso repsitorio remoto origin.

8 Resumo dos comandos no linux

```
$ git clone -> Copia um repositorio remoto para a atual localização.
$ git add <Arquivo> -> Adiciona o arquivo para ser commitado.
$ git commit -> Comita as atuais modificações para o seu repositorio local.
$ git push -> Mandas os atuais commits para o repositorio remoto.
$ git pull -> Puxa os commits do repositorio remoto.
$ git rm -> Remove um arquivo.
$ git mv -> Move um arquivo.
$ git diff -> Diferença entre os arquivos que serão comitados e suas modificações.
$ git status -> Mostra o status de cada arquivo.
$ git branch <Nome-do-Branch> -> Cria um branch com o nome passado.
$ git checkout <Nome-do3-Branch> -> Vai para o branch com o nome passado.
```

9 Git no Windows usando o tortoise git

Primeiro precisamos instalar os seguintes arquivos:

Tortoise Git	http://code.google.com/p/tortoisegit/
msysgit	http://code.google.com/p/msysgit/
PuTTY	http://www.chiark.greenend.org.uk/ sgtatham/putty/download.html

Baixe as versões mais novas, e instale tudo normalmente, sem mudar nenhuma opcao, a nao ser que seja de sua preferencia. Agora, precisamos configurar uma chave de ssh para que se computador possa con versar com o repositorio remoto. Va ate a pasta em que o PuTTY foi instalado e abra o executavel puttygen. Clique em Generate, depois digite uma senha no campo Key passphrase (essa senha sera pedida toda vez que voce fizer um push ou um pull. Nao a esqueca!). Clique em save private key, e a salve num lugar onde voce possa achar facilmente. Abra o site do github e na secao Account settings, va em SSH Public Keys e adicione a chave de ssh que voce acabou de gerar. Quando voce for clonar um repositorio, o tortoise vai pedir o seu nome e seu email. Isso e para que cada commit foito por voce tenha as suas informacoes, assim o grupo pode saber quem fez o que. Depois, voce tera que fornecer o link da chave de ssh para o tortoise, sem isso ele nao pode clonar um repositorio.