GIT Tutorial Para Iniciantes – O que é GIT?

Em 2005, Linus Torvalds (o homem conhecido por criar o núcleo, ou kernel, do SO Linux) desenvolveu o GIT, que desde então tem sido ativamente mantido por Junio ​​Hamano, um engenheiro de software japonês.

Atualmente, o GIT é um dos mais famosos sistemas de controle de versão de código aberto e milhões de projetos no mundo inteiro o utilizam para seu controle de versão (incluindo projetos comerciais e de código aberto).

O Git é totalmente grátis e pode ser instalado em Mac, Linux, Windows e Solaris diretamente do site oficial . Algumas das características essenciais do GIT são:

Um sistema de controle de versão distribuído, o GIT segue uma abordagem peer to peer, contrário de outros como o Subversion (SVN) que segue um modelo baseado em cliente-servidor.

GIT permite aos desenvolvedores ter uma infinidade de ramos de código completamente independente. Criação, exclusão e fusão desses ramos é simples e não leva tempo.

No GIT, todas as operações são atômicas. Isso significa que uma ação pode ter sucesso ou falhar (sem fazer nenhuma alteração). Isso é importante porque em alguns sistemas de controle de versão (como o CVS) onde as operações não são atômicas, se uma operação de repositório é suspensa, ela pode deixar o repositório em um estado instável.

No GIT, tudo é armazenado dentro da pasta .git. Isso não é o mesmo em outros VCS como SVN e CVS onde os metadadados de arquivos são armazenados em pastas ocultas (por exemplo, .cvs, .svn, etc.)

GIT usa um modelo de dados que ajuda a garantir a integridade criptográfica de qualquer coisa presente dentro de um repositório. Cada vez que um arquivo é adicionado ou um commit é feito, suas somas de verificação são geradas. Da mesma forma, eles são recuperados através de suas somas de verificação também.

Outra característica presente no GIT é sua área de teste ou índice. Na área de preparação, os desenvolvedores podem formatar commits e receber feedback ​​antes de aplicá-los.

O GIT é consideravelmente simples de usar. Para começar, você pode criar um repositório ou conferir um já existente. Após a instalação, um simples git-init irá deixar tudo pronto. Da mesma maneira, o comando git clone pode criar uma cópia de um repositório local para um usuário.

Fluxo de trabalho – Tutorial Git

Agora que o repositório está pronto, vamos falar sobre a estrutura que é mantida pelo GIT.

Cada repositório local consiste em três árvores: o diretório de trabalho que contém os arquivos reais; O índice que desempenha o papel de uma área de teste e o HEAD que é um ponteiro para o último commit feito pelo usuário.

Então, é assim que o fluxo de trabalho pode ser explicado: o usuário adiciona um arquivo ou alterações do diretório de trabalho para o índice (a área de teste) e uma vez revistos, o arquivo ou as alterações são finalmente comprometidos com o HEAD.

Os comandos Add e Commit:

Alterações ou adições de arquivos propostas são adicionadas ao índice usando o comando add. Para adicionar qualquer arquivo, o comando é:

git add <nome\_do\_arquivo>

Se você está realmente confiante o suficiente para fazer essas mudanças no HEAD, então você pode usar o comando commit:

git commit –m “Adicionar qualquer mensagem sobre o commit aqui”

Nota: Uma vez que o comando commit é executado (a partir do diretório de trabalho), o arquivo fica comprometido com o HEAD, mas ainda não é enviado para o repositório remoto.

Dando continuidade com as mudanças

Depois de confirmar as alterações (e acreditar que elas estão prontas para serem enviadas para o repositório original), você pode usar o comando push.

Uma vez que o git push origin master é executado de dentro do diretório de trabalho, as mudanças presentes no HEAD são enviadas para o repositório remoto. No comando acima mencionado, o master pode ser alterado para o nome do branch ao qual você deseja que as alterações sejam comprometidas.

Se, no entanto, um repositório existente ainda não tiver sido clonado e você pretende estabelecer uma ligação entre o seu repositório e um servidor remoto, execute o seguinte comando:

git remote add origin <servidor>

Nota: Substitua <servidor> pelo endereço do servidor remoto.

Uma vez clonado, quaisquer alterações feitas serão aplicadas para o servidor pertinente.

Branches

Outra característica brilhante (mas avançada) do GIT é sua capacidade de permitir que desenvolvedores e gerentes de projeto criem vários ramos (branches) independentes dentro de um único projeto.

O objetivo principal de um branch é desenvolver novas funcionalidades, mantendo-os isolados uns dos outros. O branch padrão em qualquer projeto é sempre o master branch. Tantos branches quanto necessários podem ser criados e eventualmente mesclados ao master branch.

Um novo branch pode ser criado usando o seguinte comando:

git checkout -b feature\_n \*

feature\_n é o nome do branch.

Se você deseja retornar ao master branch, o seguinte comando pode ser usado:

git checkout master

Qualquer branch pode ser excluído usando o seguinte comando:

git checkout -b feature\_n

Para tornar o branch disponível para outros usuários, você terá que movê-lo para o repositório remoto. Para fazer isso, use o seguinte comando:

git push origin feature\_n

Atualizando e dando merge

Caso você queira atualizar seu diretório de trabalho local para o mais recente do repositório remoto, o simples comando git pull pode ser usado.

Para mesclar outro branch (dar um merge) no atualmente ativo, use: git merge feature\_n.

Se você der um merge ou pull, o GIT sempre tenta lidar com os conflitos por conta própria, mas as vezes não consegue. Em caso de falha devido a conflitos, o usuário tem que resolver os conflitos manualmente. Depois de editar os arquivos (para erradicar conflitos), marque-os como merged usando:

git add <nome.arquivo>

Se antes do merge você desejar visualizar as alterações, o seguinte comando pode ser executado:

git diff <nome\_branch\_origem> <nome\_branch\_alvo>

Tagging

Antes de lançar atualizações/alterações de software, é sempre recomendado criar tags. Para fazer isso no GIT, use o seguinte comando:

git tag 1.1.0 1c2d2d56fa

O 1c2d2d56fa no comando acima refere-se aos primeiros 10 caracteres do commit-id que é referenciado com a tag. O ID de commit pode ser encontrado no log.

Log

O histórico do repositório pode ser estudado através do log. O comando git log recupera as informações. Para recuperar os commits feitos por um único usuário, você pode usar:

git log --author =Smith

Uma versão compactada do log (um commit por linha) pode ser visualizada usando:

git log --pretty=oneline

Para exibir somente os arquivos que foram alterados:

git log --name-status

Substituindo alterações locais

Se você acabou fazendo bagunça e precisa reverter as alterações feitas em qualquer arquivo, faça isso usando o seguinte comando:

git checkout -- <nomedoarquivo>

Isso substituirá as alterações da árvore de trabalho pelos últimos dados presentes no HEAD. Quaisquer alterações que já tenham sido adicionadas ao índice não serão prejudicadas.

Por outro lado, se todas as alterações/commits locais devem ser eliminados e o master branch local for necessário para apontar para o histórico mais recente do servidor, execute os seguintes comandos:

git fetch origin

git reset --hard origin/master

Conclusão – Tutorial Git