git-start

- Descrição: Guia básico de utilização do sistema de controle de versão Git.
- Versão do Guia: 1.0
- Escrito por: Fernando A. Damião <me@fadamiao.com>
- Escrito em: 2012-11-12 por Fernando A. Damião
- Última alteração: 2013-07-18 por Fernando A. Damião
- Licença: Creative Commons Attribution-ShareAlike 3.0 Unported License.

Este guia tem o intuito de demonstrar a instalação e utilização básica do sistema de controle de versão de códigos Git.

Tudo o que for demonstrado aqui pode ser utilizado em qualquer sistema operacional com Git instalado, mas mantendo o foco para utilização interna do Git, portanto não será abordada a utilização em sites como GitHub e BitBucket.

Vale lembrar que esse guia apenas demonstra funcionalidades básicas, ou seja, estão incluídos e explicados comandos com foco para o dia-a-dia.

OBS: No Git não existem papeis fixos, ou seja, não existem clientes e servidores definidos, pois o Git acaba dependendo muito mais do protocolo que será utilizado para a transferência de dados (Exs. HTTP, SSH) que uma hierarquia.

Mas para entendermos melhor serão utilizados os termos cliente e servidor, sendo o cliente uma máquina que consome repositórios e servidor uma máquina que disponibiliza repositórios.

Para aproveitar melhor e evitar problemas opte por centralizar os repositórios em uma única máquina.

Índice

Passos para instalar o Git

Instalando o Git no Windows

Primeiro, baixe o Git do site oficial.

Considerando que o Git foi baixado, faça uma cópia do seu 'find.exe' e 'sort.exe' que estão localizados, provavelmente, em 'C:\system32'.

Feito isso, inicie o instalador e siga os passos abaixo:

```
-> Welcome to the Git Setup
Next >
-> Information
Next >
-> Select Components
Next >
-> Adjusting your PATH environment
Selecione a opção 'Run Git and included UNIX tools from the Windows
Command Prompt'
Next >
-> Choosing the SSH executable
Selecione a opção 'Use OpenSSH'
Next >
-> Configuring the line ending conversions
Selecione a opção 'Checkout Windows-style, commit Unix-style line
endings'
Next >
```

Instalando o Git no Linux (Debian e derivados)

Entre no terminal como um usuário normal e digite:

```
$ sudo apt-get install git git-core
```

Configurando o ambiente

Com o Git instalado precisamos definir algumas coisas para a sua utilização, como nome e email que aparecerão nos commits, e caso seja feita a utilização de transferência de arquivos via SSH, uma chave RSA.

OBS: Esses passos devem ser feitos independente do sistema operacional utilizado.

Nome e email

Configure o nome e o email com o comando 'git config', eles aparecerão publicamente em commits, que servem para indicar quem fez as alterações, falaremos mais sobre o commit adiante.

```
$ git config --global user.name "Seu nome"
$ git config --global user.email "seunome@email.com"
```

Para conferir se está tudo configurado corretamente.

```
$ git config --global user.name
$ git config --global user.email
```

Caso queira apagar as configurações.

```
$ git config --global --unset user.email
$ git config --global --unset user.name
```

OBS: Todas as configurações feitas ficam salvas em um arquivo chamado '.gitconfig' no diretório home de seu usuário.

Chave SSH

Um dos modos mais comuns de utilizar Git é via SSH, para facilitar a utilização, ou seja, não ter que digitar a senha a todo o momento, gere uma chave SSH.

```
$ ssh-keygen -t rsa -C "Local Server"
```

A chave pode ser gerada com uma senha simples ou até em branco, de a preferência da utilização com senha.

O parâmetro '-C "Local Server"' é um comentário, utilize para indicar onde será utilizada essa chave.

Depois de gerada a chave pegue a chave pública, provavelmente o arquivo 'id_rsa.pub' dentro do diretório '.ssh' localizado no seu diretório home, e envie para o administrador do servidor onde está o Git.

Usando o Git

Com o ambiente já preparado, podemos (finalmente) começar a utilizar o Git.

Apesar de existirem diversos clientes gráficos somente abordaremos a utilização via linha de comando.

OBS: Todos os comandos devem ser feitos dentro do diretório do repositório desejado.

Lidando com repositórios

No lado do servidor

Existem duas maneiras de lidar com repositórios, no lado do cliente e no lado do servidor.

Para que toda a equipe possa utilizar o mesmo repositório, precisamos iniciar o repositório no servidor.

```
$ mkdir repositorio
$ cd repositorio
$ git init --bare
```

O comando 'git init --bare' criará a estrutura de diretórios e arquivos responsáveis por abrigar os arquivos.

No lado do cliente

No cliente o processo de inicialização é bem similar:

```
$ mkdir repositorio
$ cd repositorio
$ git init
```

OBS: Os comandos 'mkdir' e 'cd' não pertencem ao Git.

Lidando com arquivos

Após todas as alterações feitas precisamos enviá-las.

O fluxo de envio é basicamente o seguinte:

- add/rm
- commit
- push

Após cada alteração (add/rm), precisamos detalhar o que foi feito/porque foi feito, e isso é feito com um commit, após isso podemos enviar as alterações com um push.

Adicionando arquivo

Para um arquivo ser incluído utilize o comando 'git add' e nome do arquivo.

```
$ git add README
```

Removendo arquivo

Para um arquivo ser removido utilize o comando 'git rm' e nome do arquivo.

```
$ git rm README
```

Deixando uma mensagem

A cada inclusão/alteração/exclusão de arquivos, existe a necessidade de especificar o que foi feito, para isso utilizamos o comando 'git commit'.

```
$ git commit -m "Mensagem"
```

O parâmetro -m é utilizado para quando queremos descrever um commit na própria linha de comando.

Caso necessite de um maior detalhamento apenas utilize o comando 'git commit' e o editor de textos padrão de seu usuário será aberto.

OBS: O comando 'git commit' só funciona após alguma alteração no repositório (add/rm).

Enviando alterações

Depois de feitas às alterações e comentadas, pode ser feito o envio para o servidor com o comando 'git push' seguida da rota.

```
$ git push origin master
```

Lidando com rotas

Aprendemos anteriormente como enviar alterações para o servidor, mas podemos ter diferentes rotas de envio.

Adicionando uma rota de envio

Com o exemplo do comando 'git init', no lado do cliente, precisamos definir para onde irão os arquivos com o comando 'git remote add'.

```
$ git remote add origin git@192.168.0.100:/home/webmaster/repositorio
```

Como não existe nenhuma rota no exemplo, ela será criada como master.

Renomeando uma rota

Caso seja necessário renomear uma rota utilize o comando 'git remote rename' rota_antiga rota_nova.

```
$ git remote rename origin destination
```

OBS: Esse comando somente renomeia a rota.

Trocando a URL de uma rota

Caso seja necessário trocar a URL de destino utilize o comando 'git remote set-url' rota nova_url.

```
$ git remote set-url origin git@192.168.0.100:/home/webmaster/nova_url
```

Removendo uma rota

Caso não seja mais necessária à rota utilize o comando 'git remote rm' rota.

```
$ git remote rm origin
```

Listando rotas

Para verificar todas as rotas do repositório utilize o parâmetro -v no comando 'git remote'.

```
$ git remote -v
```

Obtendo repositórios

Para obter um repositório basta utilizar o comando 'git clone' seguido da URL.

```
$ git clone git@192.168.0.100:/home/webmaster/repositorio
```

Lidando com Branches

Um branch auxilia no momento de criar, testar ou corrigir funcionalidades.

Pensando no repositório como uma árvore, um branch seria um ramo dessa árvore, mas funcionando de forma independente, o que é ideal para coisas mais experimentais.

Criando um novo branch

Para criar um branch utilize o comando 'git branch' e informe o nome do branch.

```
$ git branch testes
```

Mudando de branch

Para mudar de branch, utilize o comando 'git checkout' e o branch desejado.

```
$ git checkout master
```

Criar e mudar de branch

Para criar e automaticamente mudar de branch.

```
$ git checkout -b testes
```

O parâmetro '-b' no comando 'git checkout', verifica a existência do branch 'testes' e muda para ele, caso não exista ele o cria.

Atualizando o repositório

Após baixarmos um repositório para a máquina, ele provavelmente receberá modificações, ou seja, ele não é atualizado automaticamente.

Para atualizar o repositório precisamos de dois comandos 'git fetch' e 'git merge' O comando 'git fetch' baixa as alterações, mas não as mescla no repositório, para mesclarmos as alterações precisamos do comando 'git merge'.

```
$ git fetch
$ git merge origin/master
```

Para não ter que ficar digitando dois comandos todas às vezes, basta utilizar o comando 'git pull'.

O comando 'git pull' executa sozinho, e de uma vez, os comandos 'git fetch' e 'git merge', atualizando o repositório.

```
$ git pull
```

OBS: Cuidado com conflitos com alterações que não foram enviadas.

Visualizando alterações

Você pode visualizar as alterações dos repositórios com o comando 'git log'.

```
$ git log
```

OBS: Essa forma exibe todos os commits de forma detalhada, incluindo autor e email do commit.

Para uma exibição mais simples, ou seja, somente com a mensagem do commit utilize o parâmetro '--oneline'.

```
$ git log --oneline
```

Pedindo ajuda

Quando nos esquecemos de algum comando ou parâmetro o git possui manuais próprios, basta utilizar o comando 'git help'.

```
$ git help
```

Quando a dúvida é relacionada a um comando em específico, basta informar o comando.

```
$ git help commit
```

OBS: Quando o comando é executado em uma máquina com Windows o manual irá aparecer no navegador padrão da máquina, se executado em Linux/Mac OS X ele abrirá uma man-page normalmente.

Resumo Geral

O resumo demonstra como operar um repositório, adicionar um arquivo, adicionar um commit, e enviar as alterações em dois cenários um sem existir o repositório e o outro existindo o repositório.

Repositório não existe

Faça esses comandos no servidor:

```
$ mkdir repositorio
$ cd repositorio
$ git init --bare
```

Faça esses comandos no cliente:

```
$ mkdir repositorio
$ cd repositorio
$ touch README
$ git init
$ git add README
$ git commit -m "Inclusão do README"
$ git remote add origin git@192.168.0.100:/home/webmaster/repositorio
$ git push -u origin master
```

Repositório já existe

Comandos apenas para o cliente:

```
$ git clone git@192.168.0.100:/home/webmaster/repositorio
$ git add README
$ git commit -m 'Comentário sobre inclusão/atualização/modificação
feita'
$ git push -u origin master
```

Links de ajuda

Os links abaixo são guias mais detalhados para utilização do Git:

- Easy Version Control with Git em en_US
- Git Immersion em en_US - Passo-a-Passo sobre como utilizar Git.
- git ready em en_US
- github:help em en_US - Foca a utilização do Git no site GitHub, mas possui guias/tutoriais que auxiliam em utilizações fora do site.
- Git from the bottom up em en_US - Guia bem detalhado sobre Git.
- Rapid7 - Git Cheatsheet em en_US

Referências

Além dos links de ajuda, esses links ajudam a compreender mais sobre o assunto e ajudaram a escrever esse guia.

- Site oficial do Git em en_US
- Git Reference em en_US
- git - guia prático em pt_BR
- Apostila de introdução ao controle de versões com Git em pt_BR
- git cheat sheet em en_US

Vale a pena ver

Para continuar os estudos, cadastre-se nesses sites e suba os seus repositórios:

- GitHub em en_US
- Bitbucket em en_US

Para aprender passo a passo:

- Try Git em en_US
- Git Immersion em en_US
- Git Real em en_US
- Git Real 2 em en_US

Conclusão

Sem dúvida nenhuma o Git é uma ferramenta poderosa, todos os comandos demonstrados nesse guia são para auxiliar no dia-a-dia da utilização do Git.

Obviamente existem diversos comandos e parâmetros que não foram citados aqui, mas esse não é o intuito deste guia.

O intuito é dar uma noção um pouco além da básica para a operação desse sistema de controle de versão.