Seminário sobre GIT

Bruno César Ribas

Centro de Computação Científica e Software Livre

21 de outubro de 2007

Sumário

O que é?!

GIT é um sistema de controle de revisão distribuído, com uma vasta gama de comandos que provêem comandos de alto nível como de acesso completo na parte interna do sistema

E o que significa GIT?

De acordo com *Linus Torvalds* depende do seu humor:

- 'git' é uma gíria Britânica para "silly, stupid, or worthless person"
- 3-letras aleatórias que é pronunciável, e não é usado por algum comando UNIX comumente usado.
- "Global Information Tracker": quando está em bom humor e está funcionando "Angels sing, and a light suddenly fills the room."
- "Goddamn idiotic truckload of sh*t": quando quebra

E o que significa GIT?

De acordo com *Linus Torvalds* depende do seu humor:

- 'git' é uma gíria Britânica para "silly, stupid, or worthless person"
- 3-letras aleatórias que é pronunciável, e não é usado por algum comando UNIX comumente usado.
- "Global Information Tracker": quando está em bom humor e está funcionando "Angels sing, and a light suddenly fills the room."
- "Goddamn idiotic truckload of sh*t": quando quebra

Características Únicas

- Idéia básica de desenvolvimento não linear. Facilidade de BRANCH e MERGE
- Desenvolvimento Distribuído. cada desenvolvedor "ganha"uma cópia local que é a arvore inteira de desenvolvimento.
- Repositórios podem ser facilmente publicados via: HTTP, FTP, ssh, rsync, git (o protocolo próprio).
- Emulador de Servidor CVS (e futuramente de SVN), o que facilita algumas integrações
- Desenvolvido seguindo a tradição UNIX, vários códigos primitovs em C com ferramentas (scripts) para facilitar o uso
- Vários Algoritmos para execução de Merge

Sumário

Iniciando

Para existir uma boa relação de desenvolvimento é bom de fato conhecer a equipe e as ferramentas.

Apresentando-se ao GIT

```
$ git config --global user.name "Meu Nome"

$ git config --global user.email meu-email@c3sl.ufpr.br
```

Importando um novo projeto

Imagine a existência de um projeto qualquer: `chessd.tar.bz2`
E ainda, esse projeto não está sobre um bom controle de versão!

Colocando em controle de versão

```
$ tar xjf chessd.tar.bz2
$ cd chessd
$ git init
Initialized empty Git repository in .git/
$ #adicionando o conteúdo
$ git add .
$ # e finalmente
$ git commit
```

Fazendo alterações

Certo! Agora temos um projeto em controle de versão! Vamos trabalhar em cima.

Modificando

```
$ vi novoarquivo.c
##gerando codigo aleatorio##
$ vi chessd.c
###modificando, totalmente independente do novoarquivo###
$ git diff
#receberá o diff de tudo modificado
$ git add novoarquivo.c
#adiciona o novo arquivo nocontrole
$ git commit novoarquivo.c
#o commit pedira uma descrição da alterção
$ git commit chessd.c
#o commit pedirá uma descrição da alteração
```

Política de COMMIT

- Alterações não relacionadas gera commits separados
  - Alterei o suporte a SQL <- COMMIT A
  - Alterei o controle de fluxo do socket <- COMMIT B
- Alterações não relacionadas no mesmo arquivo?!?!
  - É excelente fazer um commit quando terminar uma classe de alteração.
  - Antes de fazer uma alteração não relacionada a anterior, fazer o commit antes.

Olhando o Changelog

Consideramos as mensagens geradas pelos commits como o changelog do projeto.

Olhando o Changelog

```
$ git log
#ver os logs desde o início até agora
$ git log -p

#alem do log ver os diffs de cada passo
$ git log v1.0.0 v1.0.2

#ver o log entre a versão 1.0.0 e 1.0.2
```

Tagging

Tag é marcar uma revisão com algum nome específico.
Geralmente Tag é usada para identificar uma versão estável do desenvolvimento.

Criando Tag - Leve

Uma Tag Leve é usada só para o controle do programador. Facilitador de indicador de problemas

```
$ git tag minha-tag
#pronto! Tag criada
```

Criando Tag - "annotated"

Usada para demarcar o território (como uma nova versão)

```
$ git tag -s minha-tag
#necessário entrar com a descrição da Tag, além de pedir uma ch
$ git tag -a minha-tag
#igual a anterior, porém não assina.
```

Branches - conceito

- Duplicação de um objeto em controle de revisão. Onde o novo objeto criado tem o mesmo conteúdo da versão original.
- Normalmente Branch implica em MERGE, que é o processo de aplicar as diferenças do branch em outro (normalmente de onde se originou).
- idéia boa quando vai manter duas versões de um projeto
  - Branch MASTER (default) é a cabeça de desenvolvimento
  - Branch STABLE seria o Branch da TAG estável, e apenas se corrigem bugs (sempre faz merge com a MASTER)

Branch

Trabalhando com Branch

```
$ git branch
  experimental
  * master
#mostra os branches existestes, e o "*" é que está na árvore
$ git checkout experimental
(editar aquivos)
$ git commit <arquivos>
$ git checkout master
#alterações não são visiveis no master
$ git merge experimental
#aplica alterações do experimental no master
```

Criando Branch

Criando Branch

```
$ git checkout -b meu-branch
# cria um branch baseado na arvore que está usando

$ git checkout -b meu-branch minha-tag
#cria um branch com a árvore até marcada com minha-tag

$ git branch -d meu-branch
#remove o branch e garante que tenha sido feito o merge com
#o master

$git branch -D experimental
#remove o branch e não verifica se aplicou o merge com o
#master
```

Trabalhando com Colaboração

Um dos propósitos de manter um projeto em controle de versão é também a sua facilidade de manter um grupo de trabalho atualizado.

repositório de colaboração

```
$ git clone git@shiva:chessd/chessdserver.git
#isso cria um diretório chessdserver em $(pwd) com a árvore
#completa de desenvolvimento desse projeto

$ git pull
#atualiza o repositório em relação ao guardado no servidor
#faz eventuais merges em relação a sua árvore

$ git push
#envia toda a sua pilha de commits para o servidor

$ git push --tags
#envia a sua pilha de Tags criadas para o servidor
```

Achando Bugs introduzidos

É comum acontecer a entrada de um bug inesperada em nosso sistema. Para isso o GIT fornece uma ferramenta que faz um busca binária a procura do bug

Achando bugs

```
$ git bisect start
$ git bisect bad
#diz que a árvore atual está ruim
$ git bisect good v1.0.0
#avisa que a tag v1.0.0 é a versão que com certeza
#está boa
```

Sumário

Referências

http://git.or.cz/

http://www.kernel.org/pub/software/scm/git/docs/

http://www.kernel.org/pub/software/scm/git/docs/tutorial.html

http://www.kernel.org/pub/software/scm/git/docs/core-tutorial.h

Sumário

Dúvidas,

Perguntas,

Choro,

Lamentações