Introdução ao Git

Anderson Moreira

Latest update: Aug 28, 2015,
Document sources, updates and translations:
http://free-electrons.com/docs/git
Corrections, suggestions, contributions and translations are welcome!

O que é o Git?

- Um sistema de controle de versão, como CVS, SVN, Perforce or ClearCase.
- Originalmente desenvolvido para o desenvolvimento do kernel do Linux, é utilizado por um grande número de projetos, incluindo o U-Boot, GNOME, Buildroot, uClibc e muitos outros.
- Ao contrário do CVS ou SVN, Git é um controle de versão distribuído.
  - Não utiliza repositório central.
  - Cada usuário tem um repositório local.
  - Ramificação local é possível e muito importante.
  - Compartilhamento fácil e simples entre desenvolvedores.
  - Bem adequado para o modelo de desenvolvimento colaborativo usado em projetos de código aberto.

Instalação

- O Git está disponível como um pacote na sua distribuição linux:
  `sudo apt-get install git-core`
- Através do comando “git” é possível acessar tudo que está no repositório.
  - `git` possui vários comandos, chamados usando: `git <comando>`, onde `<comando>` pode ser `clone`, `checkout`, `branch`, etc.
  - Ajuda pode ser encontrada no comando:
    `git help <comando>`
- Cadastre seu nome e email
  - Eles estarão referenciados em cada um dos commits.
  - `git config --global user.name 'My Name'`
  - `git config --global user.email me@mydomain.net`

Clonando um repositório

- Para começar a trabalhar em um projeto, utilize operação de clonagem do Git.
- Com CVS ou SVN, seria utilizado a operação de “checkout” para ter a última cópia da versão do projeto(mais nova).
- Com o Git, uma cópia de todo o repositório é realizada em seu computador, incluindo o histórico, que permite que sejam realizadas operações mesmo quando se está offline.
- Clonando o repositório do kernel do Linux de Linus Torvalds

```
git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git
```

- O `git://` é um protocolo especill. A maioria dos repositórios pode, também, ser acessado utilizando: `http://`, porém é relativamente mais lento.
- Depois de clonar, no linux-2.6/, é disponibilizado o repositório com a cópia funcional da branch “master.

Explorando o histórico

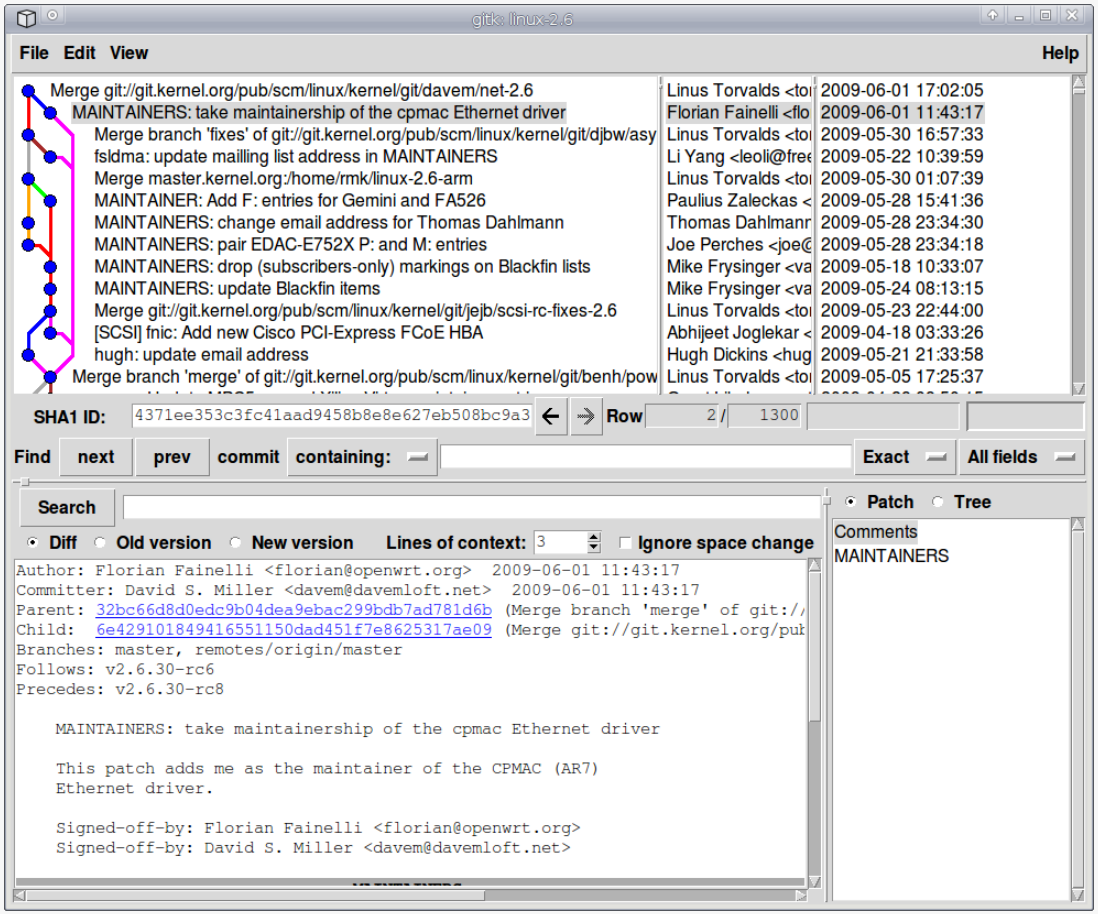
- O “`git log`” lista todos os commits. O último commit é o que aparece primeiro.

```
commit 4371ee353c3fc41aad9458b8e8e627eb508bc9a3
Author: Florian Fainelli <florian@openwrt.org>
Date:   Mon Jun 1 02:43:17 2009 -0700
    MAINTAINERS: take maintainership of the cpmac Ethernet driver
    This patch adds me as the maintainer of the CPMAC (AR7)
    Ethernet driver.
    Signed-off-by: Florian Fainelli <florian@openwrt.org>
    Signed-off-by: David S. Miller <davem@davemloft.net>
```

- `git log -p` irá listar todos os commits com o seu “diff”(acima) correspondente.
- O histórico do Git não é linear como CVS or SVN, porém é um gráfico de commits.
  - Deixa um pouco mais complicado para ser entendido no começo.
  - Porém permite que ferramentas poderosas do Git sejam utilizadas: (distributed, branching, merging)

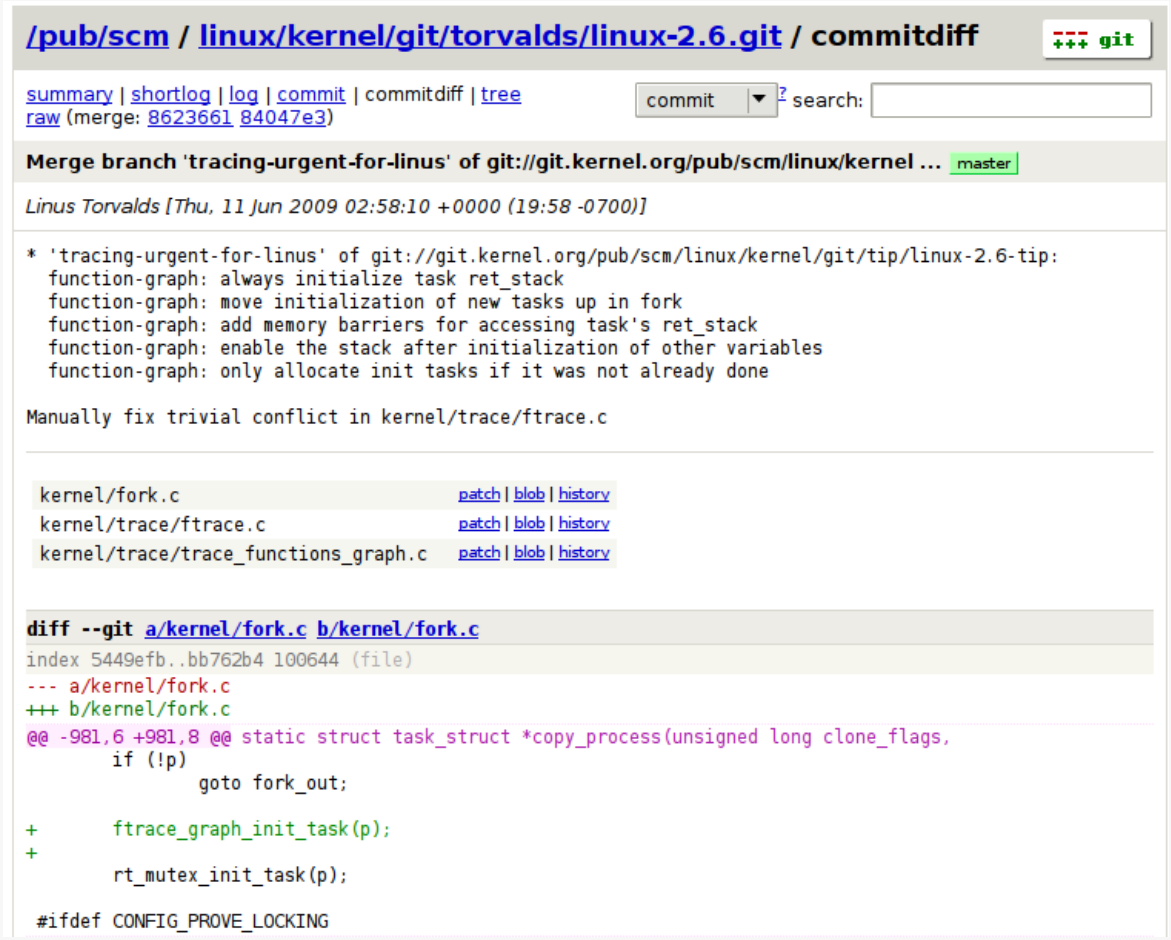
Visualizando o histórico

- *gitk* é uma ferramenta gráfica que representa o histórico do repositório Git
- Pode ser instalado diretamente do pacote `gitk`



Visualizando o histórico

- Outra ótima ferramenta é a interface Web para o Git. Para ter o kernel, ele está disponível em: http://git.kernel.org/



/pub/scm / linux/kernel/git/torvalds/linux-2.6.git / commitdiff

git

summary | shortlog | log | commit | commitdiff | tree
raw (merge: 8623661 84047e3)

commit ? search:

Merge branch 'tracing-urgent-for-linus' of git://git.kernel.org/pub/scm/linux/kernel ... master

Linus Torvalds [Thu, 11 Jun 2009 02:58:10 +0000 (19:58 -0700)]

```
* 'tracing-urgent-for-linus' of git://git.kernel.org/pub/scm/linux/kernel/git/tip/linux-2.6-tip:
  function-graph: always initialize task ret_stack
  function-graph: move initialization of new tasks up in fork
  function-graph: add memory barriers for accessing task's ret_stack
  function-graph: enable the stack after initialization of other variables
  function-graph: only allocate init tasks if it was not already done

Manually fix trivial conflict in kernel/trace/ftrace.c
```

kernel/fork.c	patch | blob | history
kernel/trace/ftrace.c	patch | blob | history
kernel/trace/trace_functions_graph.c	patch | blob | history

diff --git a/kernel/fork.c b/kernel/fork.c

```
index 5449efb..bb762b4 100644 (file)
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -981,6 +981,8 @@ static struct task_struct *copy_process(unsigned long clone_flags,
        if (!p)
                goto fork_out;

+       ftrace_graph_init_task(p);
+
        rt_mutex_init_task(p);

 #ifdef CONFIG_PROVE_LOCKING
```

Atualizando o seu repositório

- O repositório que foi clonado no começo do projeto será mudado com o tempo.
- Ao atualizar o repositório local, isto irá refletir nas mudanças realizadas no repositório remoto, e será necessário atualizar o seu repositório de tempos em tempos.
- `O comando “git pull”`
- Internamente, realiza duas coisas:
  - Busca as novas mudanças no repositório remoto (`git fetch`)
  - Funde as novas mudanças na branch atual (`git merge`)

Tags

- A lista das tags existentes podem ser achadas utilizando:
  `git tag -l`
- Para checar a versão funcional do repositório com uma tag dada:
  `git checkout <nomedatag>`
- Para mostrar a lista de mudanças entre uma determinada tag e a versão mais nova:
  `git log v2.6.30..master`
- Para mostrar a lista de mudanças com “diff" em um arquivo entre duas tags:
  `git log v2.6.29..v2.6.30 MAINTAINERS`
- Com gitk:
  `gitk v2.6.30..master`

Ramificações (Branches)

- Para começar a trabalhar em algo, é melhor criar uma ramificação (branch)
  - É apenas local, só você consegue enxergar a “branch”
  - É rápida
  - Permite que o trabalho seja dividido em diferentes tópicos, podendo ser descartado posteriormente.
  - É barato, mesmo que você pense que esteja fazendo algo rápido e fácil, crie uma “branch”
- Diferente de outros sistemas de controle de versão, Git encoraja o uso de “branches”. Não exite em utilizá-las.
- Na disciplina de SE não iremos utilizar *Branch*

Ramificações (Branches)

- Crie uma branch
  `git branch <nomedabranch>`
- Mova essa branch
  `git checkout <nomedabranch>`
- Os dois comandos anteirores ao mesmo tempo:
  `git checkout -b <nomedabranch>`
- Listar as “branches” locais
  `git branch -l`
- Listar as “branches”, includindo as branches remotas
  `git branch -a`

Realizando mudanças

- Edite um arquivo com seu editor de texto favorito
- Mostrar o status da versão do seu repositório:
  `git status`
- Git tem uma característica chamada *index*, que permite organizar os commits antes de serem “commitados. O que permite que apenas uma parte das modificações seja realizada, separada por arquivo ou por pedaço de código.
- Para cada arquivo modificado:
  `git add <nomedoarquivo>`
- Depois commit. Não há necessidade de estar online para realizar o commit:
  `git commit`
- Se todos os arquivos modificados devem estar no commit:
  `git commit -a`

Compartilhando mudanças por email

- O jeito mais simples de compartilhar algumas mudanças é mandar patches por email.
- O primeiro passo é gerar os patches
  `git format-patch -n master..<suabranch>`
  - Será gerado um patch para cada um dos commits realizados na branch escolhida `<suabranch>`
  - Os arquivos do patch serão 0001-...., 0002-...., etc.
- O segundo passo é enviar esses patches
  `git send-email --compose --to email@domain.com 00*.patch`
  - Assumindo que o sistema de email local está propriamente configurado
    Necessário: `git-email` package no Ubuntu.
  - Ou "`git config`" permite que seja configurado o servidor SMTP, porta, usuário e senha, caso necessário.

Compartilhando mudanças: Repositório

- Caso sejam feitas muitas mudanças nos arquivos e queira simplificar a colaboração com os outros contribuintes do repositório, a melhor opção é ter seu próprio repositório.
- Crie uma nova versão do seu repositório.

```
cd /tmp
git clone --bare ~/project project.git
touch project.git/git-daemon-export-ok
```

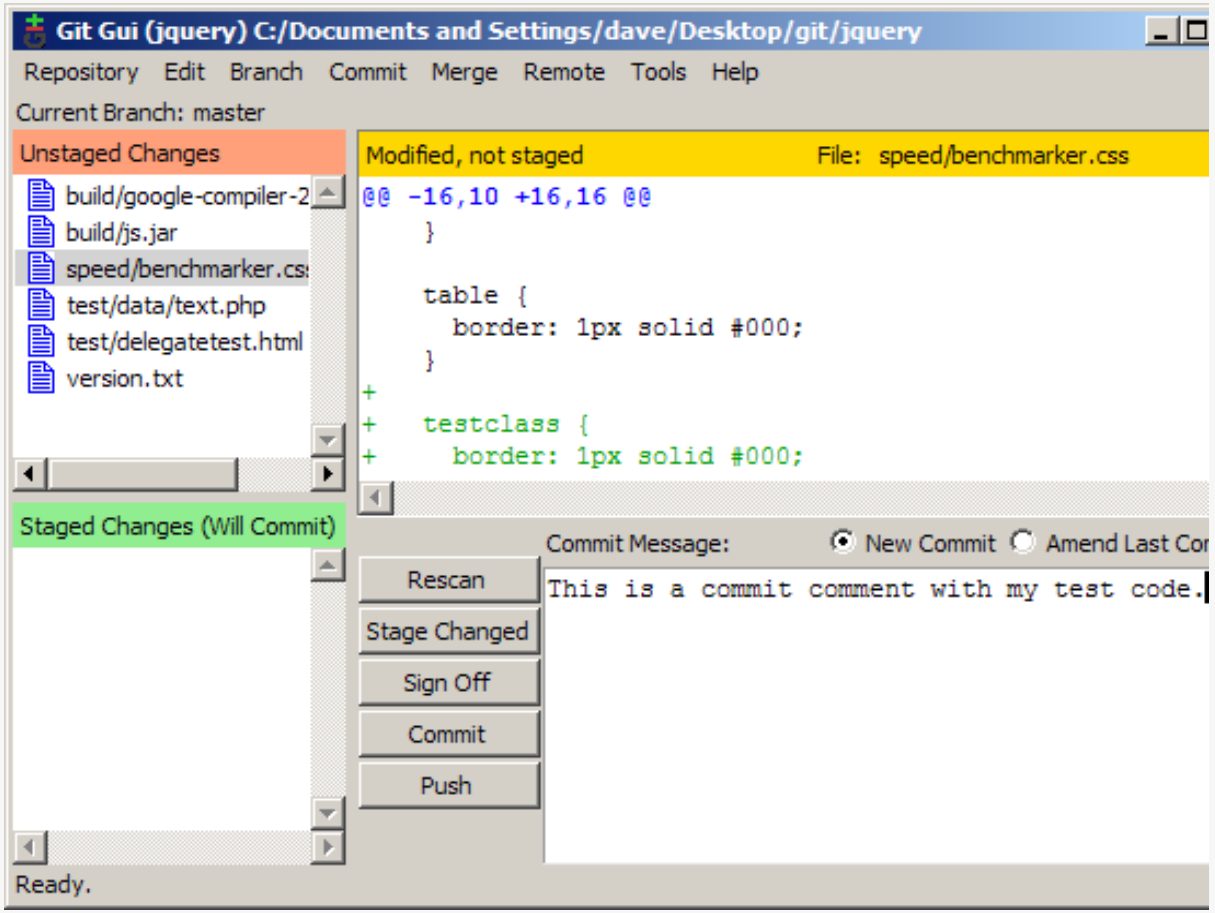
- Transfira o conteúdo de `project.git` para um lugar público (Acessível para leitura por HTTP para todos, e leitura-gravação pelo SSH)
- Clone este repositório: http://yourhost.com/path/to/project.git
- “Push” suas mudanças usando:

```
git push ssh://yourhost.com/path/toproject.git
srcbranch:destbranch
```

Monitorando árvores remotas

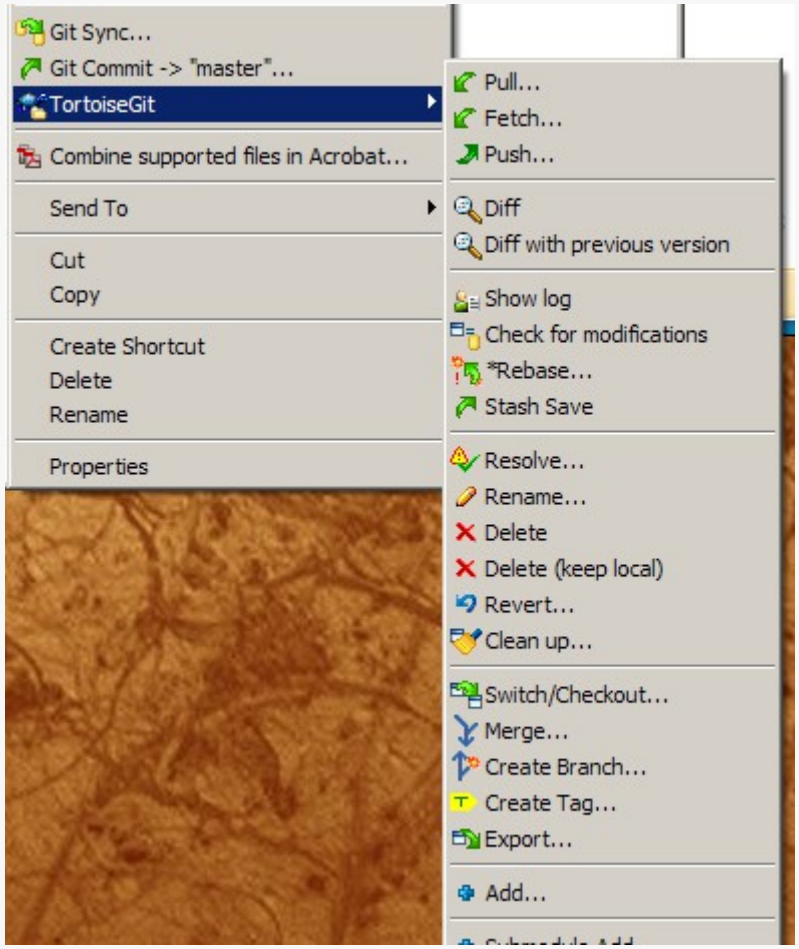
- Além da árvore oficial de Linus Torvalds , você talvez queira utilizar outros desenvolvimentos ou árvores experimentais.
  - A árvore OMAP
    ```
    git://git.kernel.org/pub/scm/linux/kernel/git/tmlind/linux-omap-2.6.git
    ```
  - A árvore de tempo real
    ```
    git://git.kernel.org/pub/scm/linux/kernel/git/rostedt/linux-2.6-rt.git
    ```
- O comando remoto de git permite gerir árvores remotas.
  ```
  git remote add rt git://git.kernel.org/pub/scm/linux/kernel/git/rostedt/linux-2.6-rt.git
  ```
- Pegue o conteúdo da árvore
  ```
  git fetch rt
  ```
- Mude para uma das branches
  ```
  git checkout rt/master
  ```

Alguns Clientes GIT



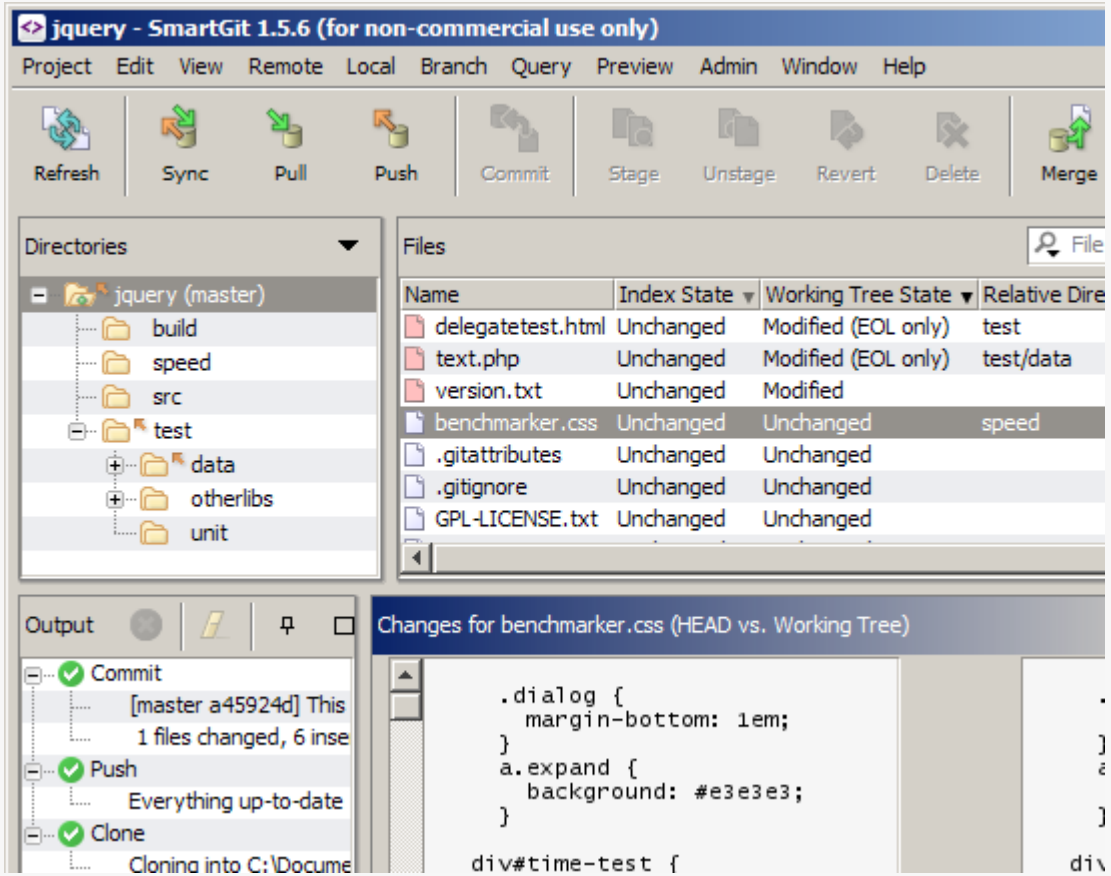
Git for Windows
https://git-for-windows.github.io/

Alguns Clientes GIT



Tortoise Git
http://tortoisesvn.net/

Alguns Clientes GIT



SmartGIT

http://www.syntevo.com/smartgit/

Sobre o Git

- Nós vimos as características básicas do Git.
  Há muito mais recursos disponíveis(rebasing, bisection, merging and more)
- Referências:
  - Git Manual
    http://www.kernel.org/pub/software/scm/git/docs/user-manual.html
  - Git Book
    http://book.git-scm.com/
  - Git official website
    http://git-scm.com/
  - James Bottomley's tutorial on using Git
    http://free-electrons.com/pub/video/2008/ols/ols2008-james-bottomley-git.ogg

Atividade Prática – Git

- Clone um repositório Git e explore o histórico.
- Crie e compartilhe mudanças para um projeto gerido em um Git.