Controle de Versão com git

Felipe Oliveira Carvalho

Universidade Federal de Sergipe

6 de Outubro, 2011

O que é controle de versão?

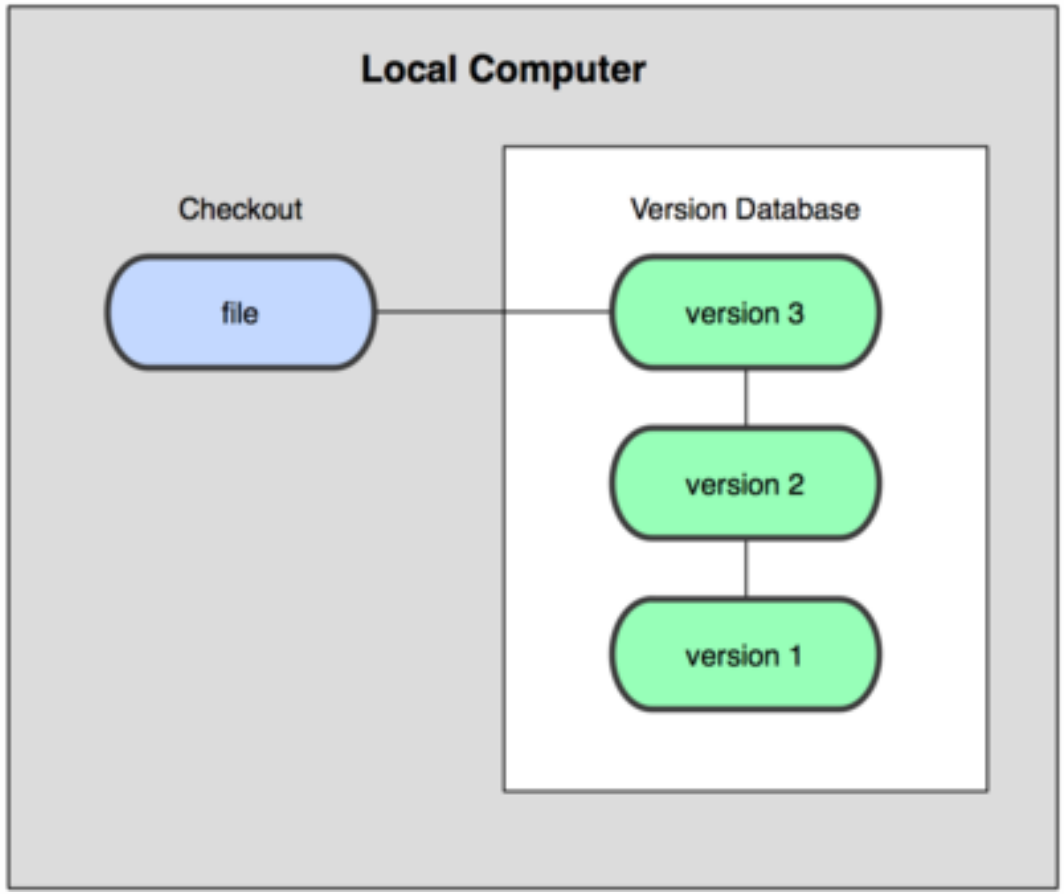
Controle de versão é um sistema que grava as mudanças feitas em um conjunto de arquivos ao longo do tempo de uma forma que você possa restaurar e comparar versões específicas depois.

O que é controle de versão?

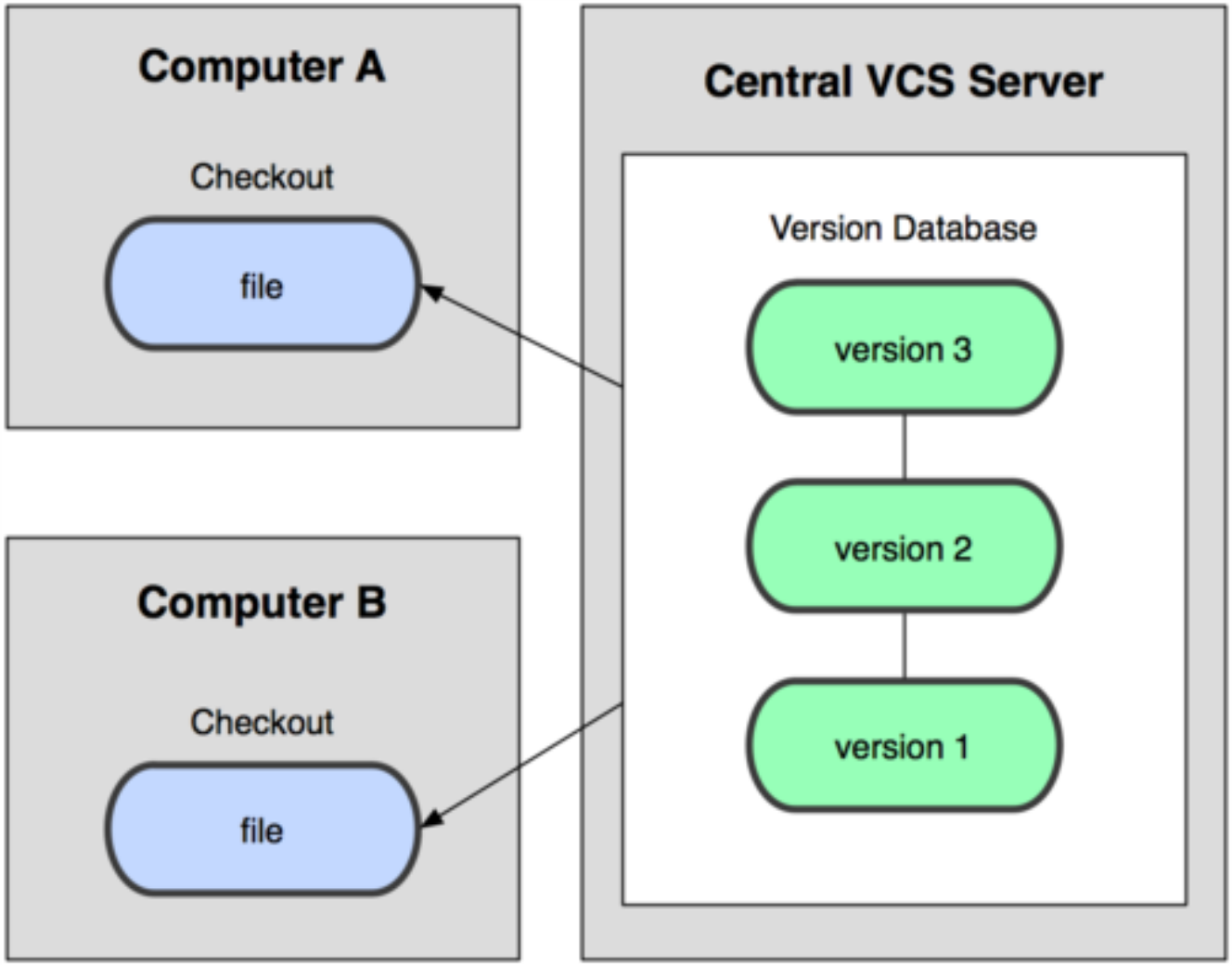
Controle de versão é um sistema que grava as mudanças feitas em um conjunto de arquivos ao longo do tempo de uma forma que você possa restaurar e comparar versões específicas depois.
Tipos de controle de versão:

- Controle de versão local
- Controle de versão centralizado
- Controle de versão distribuído

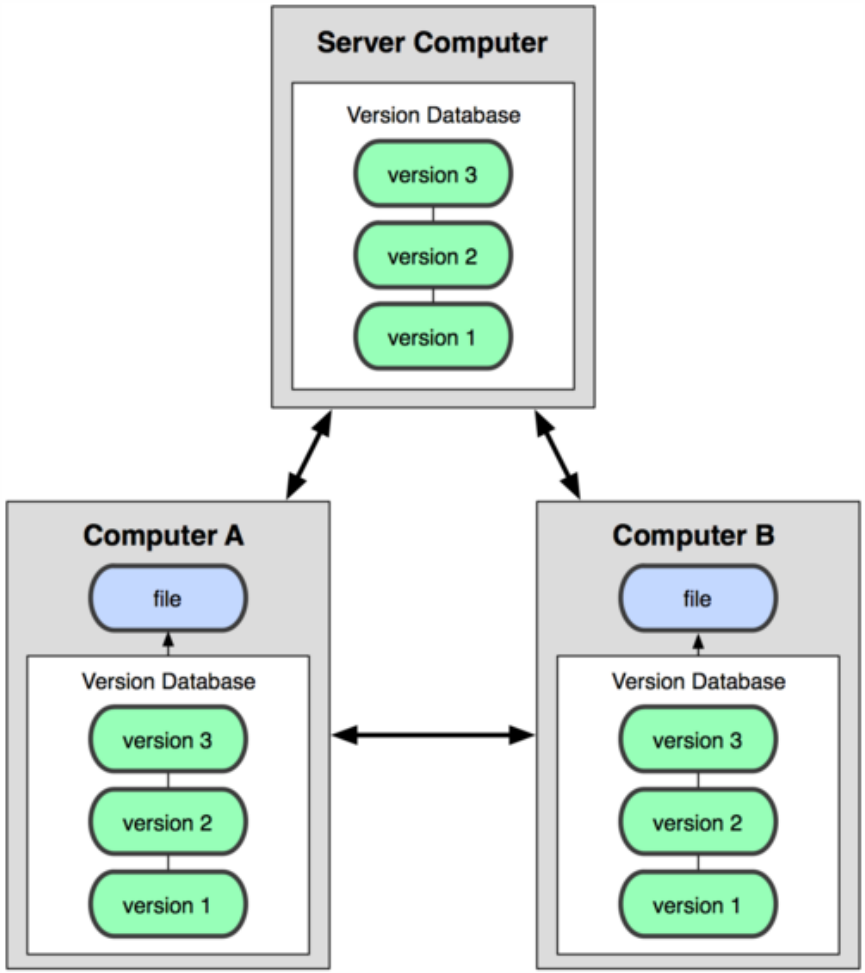
Controle de versão local



Controle de versão centralizado



Controle de versão distribuído



O que é git?

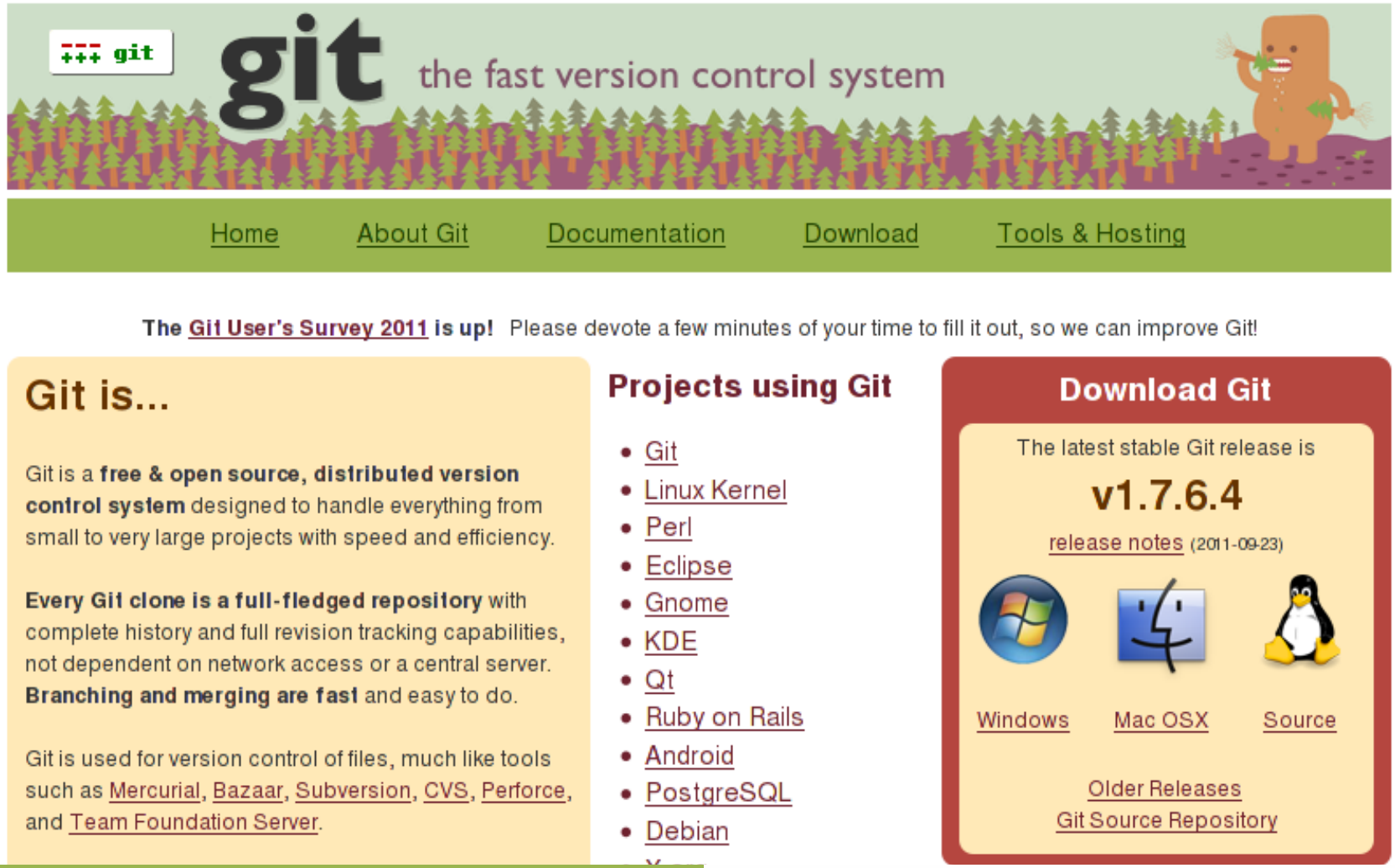
Git é um sistema de controle de versão **distribuído** projetado para ser **eficiente**.

Breve história

- Criado por Linus Torvalds em 2005
- Para ser usado no desenvolvimento do Linux kernel
- Metas dos projeto:
  - Velocidade
  - Design simples
  - Permitir desenvolvimento não-linear (milhares de branches)
  - Capaz de manipular projetos grandes como o Linux kernel de maneira eficiente

Instalação

http://git-scm.com



Configurando

```
$ git config --global user.name "Felipe O. Carvalho"
$ git config --global user.email "felipekde@gmail.com"
```

Criando um repositório

`git init`

```
$ mkdir hello_git
$ cd hello_git/
$ git init
Initialized empty Git repository in /home/felipe/hello_git/
$ ls -a
.  ..  .git
```

Primeiro commit

```
git add
git commit
```

```
$ touch hello_world.py
$ git add hello_world.py
$ git commit -m "Primeiro commit"
```

Primeiro commit

```
$ git show
```

Primeiro commit

```
$ git show

commit 9b302737d41712809ca455b1d522c334793ef001
Author: Felipe Oliveira Carvalho <felipekde@gmail.com>
Date:   Sun Oct 2 15:45:23 2011 -0300

    Primeiro commit

    diff --git a/hello_world.py b/hello_world.py
    new file mode 100644
    index 0000000..e69de29
```

Clonando um Repositório

`git clone`

```
$ git clone git://github.com/schacon/ticgit.git
```

Clonando um Repositório

git clone

```
$ git clone git://github.com/schacon/ticgit.git
```

```
Cloning into ticgit...
remote: Counting objects: 1857, done.
remote: Compressing objects: 100% (826/826), done.
remote: Total 1857 (delta 969), reused 1796 (delta 931)
Receiving objects: 100% (1857/1857), 269.46 KiB | 143 KiB/s, done.
Resolving deltas: 100% (969/969), done.
```

Clonando um Repositório

git clone

```
$ git clone git://github.com/schacon/ticgit.git
```

```
Cloning into ticgit...
remote: Counting objects: 1857, done.
remote: Compressing objects: 100% (826/826), done.
remote: Total 1857 (delta 969), reused 1796 (delta 931)
Receiving objects: 100% (1857/1857), 269.46 KiB | 143 KiB/s, done.
Resolving deltas: 100% (969/969), done.
```

```
$ cd ticgit
```

Clonando um Repositório

git clone

```
$ git clone git://github.com/schacon/ticgit.git
```

```
Cloning into ticgit...
remote: Counting objects: 1857, done.
remote: Compressing objects: 100% (826/826), done.
remote: Total 1857 (delta 969), reused 1796 (delta 931)
Receiving objects: 100% (1857/1857), 269.46 KiB | 143 KiB/s, done.
Resolving deltas: 100% (969/969), done.
```

```
$ cd ticgit
```

```
$ ls
```

Clonando um Repositório

git clone

```
$ git clone git://github.com/schacon/ticgit.git
```

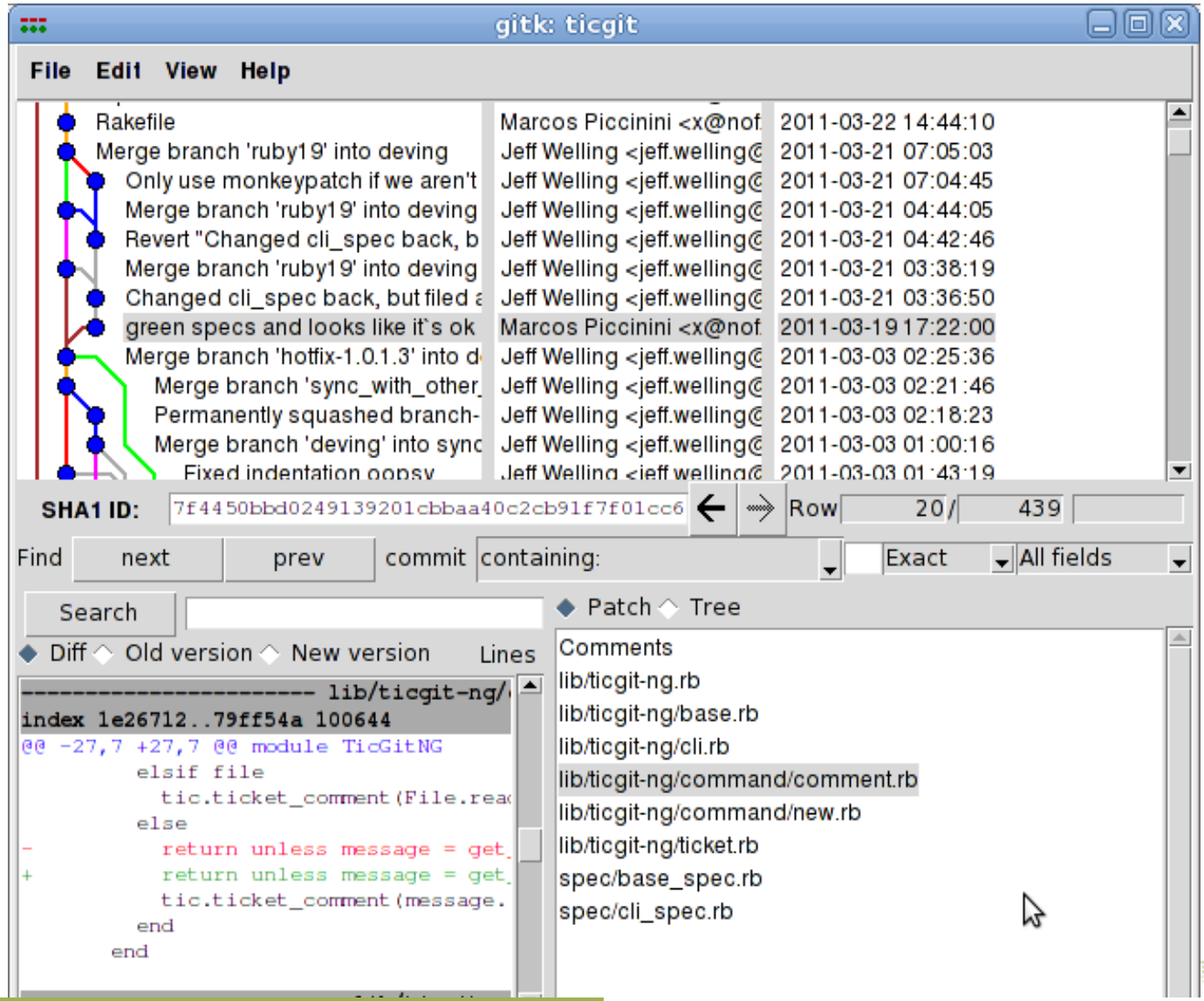
```
Cloning into ticgit...
remote: Counting objects: 1857, done.
remote: Compressing objects: 100% (826/826), done.
remote: Total 1857 (delta 969), reused 1796 (delta 931)
Receiving objects: 100% (1857/1857), 269.46 KiB | 143 KiB/s, done.
Resolving deltas: 100% (969/969), done.
```

```
$ cd ticgit
```

```
$ ls
```

```
bin  examples  lib  LICENSE_GPL  LICENSE_MIT note  Rakefile
README.mkd  spec ticgit-ng.gemspec  TODO
```

gitk


gitk: ticgit
File Edit View Help
Rakefile
Merge branch 'ruby19' into deving
Only use monkeypatch if we aren't
Merge branch 'ruby19' into deving
Revert "Changed cli_spec back, b
Merge branch 'ruby19' into deving
Changed cli_spec back, but filed a
green specs and looks like it`s ok
Merge branch 'hotfix-1.0.1.3' into d
Merge branch 'sync_with_other
Permanently squashed branch-
Merge branch 'deving' into sync
Fixed indentation oopsy
Marcos Piccinini <x@nof 2011-03-22 14:44:10
Jeff Welling <jeff.welling@ 2011-03-21 07:05:03
Jeff Welling <jeff.welling@ 2011-03-21 07:04:45
Jeff Welling <jeff.welling@ 2011-03-21 04:44:05
Jeff Welling <jeff.welling@ 2011-03-21 04:42:46
Jeff Welling <jeff.welling@ 2011-03-21 03:38:19
Jeff Welling <jeff.welling@ 2011-03-21 03:36:50
Marcos Piccinini <x@nof 2011-03-19 17:22:00
Jeff Welling <jeff.welling@ 2011-03-03 02:25:36
Jeff Welling <jeff.welling@ 2011-03-03 02:21:46
Jeff Welling <jeff.welling@ 2011-03-03 02:18:23
Jeff Welling <jeff.welling@ 2011-03-03 01:00:16
SHA1 ID: 7f4450bbd0249139201cbbaa40c2cb91f7f01cc6
Row 20/ 439
Find next prev commit containing: Exact All fields
Search
Patch Tree
Diff Old version New version Lines
index 1e26712..79ff54a 100644
@@ -27,7 +27,7 @@ module TicGitNG
elsif file
tic.ticket_comment(File.rea
else
- return unless message = get_
+ return unless message = get_
tic.ticket_comment(message.
end
end
Comments
lib/ticgit-ng.rb
lib/ticgit-ng/base.rb
lib/ticgit-ng/cli.rb
lib/ticgit-ng/command/comment.rb
lib/ticgit-ng/command/new.rb
lib/ticgit-ng/ticket.rb
spec/base_spec.rb
spec/cli_spec.rb

Workflow básico

- Editar arquivos

Workflow básico

- Editar arquivos – **Eclipse, Visual Studio, Notepad++, emacs, vim... Photoshop...**

Workflow básico

- Editar arquivos – **Eclipse, Visual Studio, Notepad++, emacs, vim... Photoshop...**
- Adicionar as mudanças ao *index* (também conhecido como *staging area*

Workflow básico

- Editar arquivos – **Eclipse, Visual Studio, Notepad++, emacs, vim... Photoshop...**
- Adicionar as mudanças ao *index* (também conhecido como *staging area* – **git add, git rm, git reset...**

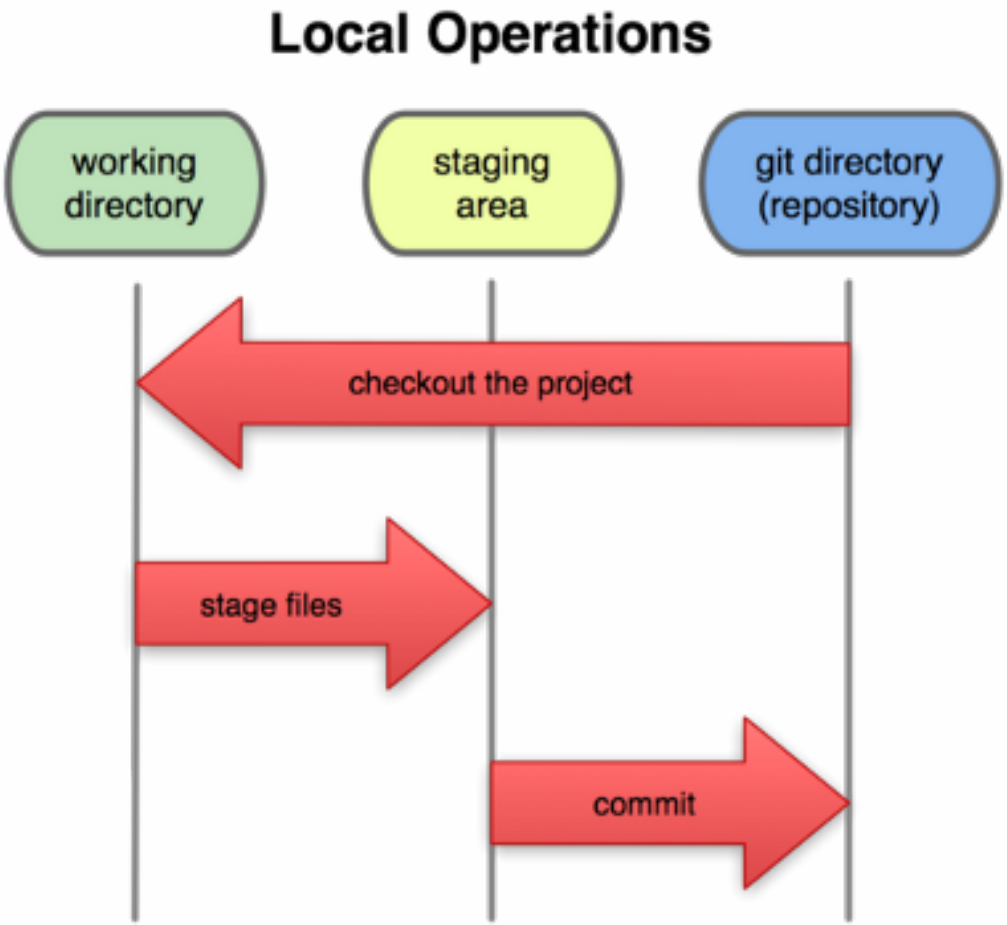
Workflow básico

- Editar arquivos – **Eclipse, Visual Studio, Notepad++, emacs, vim... Photoshop...**
- Adicionar as mudanças ao *index* (também conhecido como *staging area* – **git add, git rm, git reset...**
- Revisar as mudanças – **git status, git diff...**

Workflow básico

- Editar arquivos – **Eclipse, Visual Studio, Notepad++, emacs, vim... Photoshop...**
- Adicionar as mudanças ao *index* (também conhecido como *staging area* – **git add, git rm, git reset...**
- Revisar as mudanças – **git status, git diff...**
- Fazer o *commit* das mudanças – **git commit -m "Mensagem"**

Os três passos



Os três passos

```
$ vim hello_world.py
```

Os três passos

```
$ vim hello_world.py

$ cat hello_world.py
print "Hell World!"
```

Os três passos

```
$ vim hello_world.py

$ cat hello_world.py
print "Hell World!"

$ git status
```

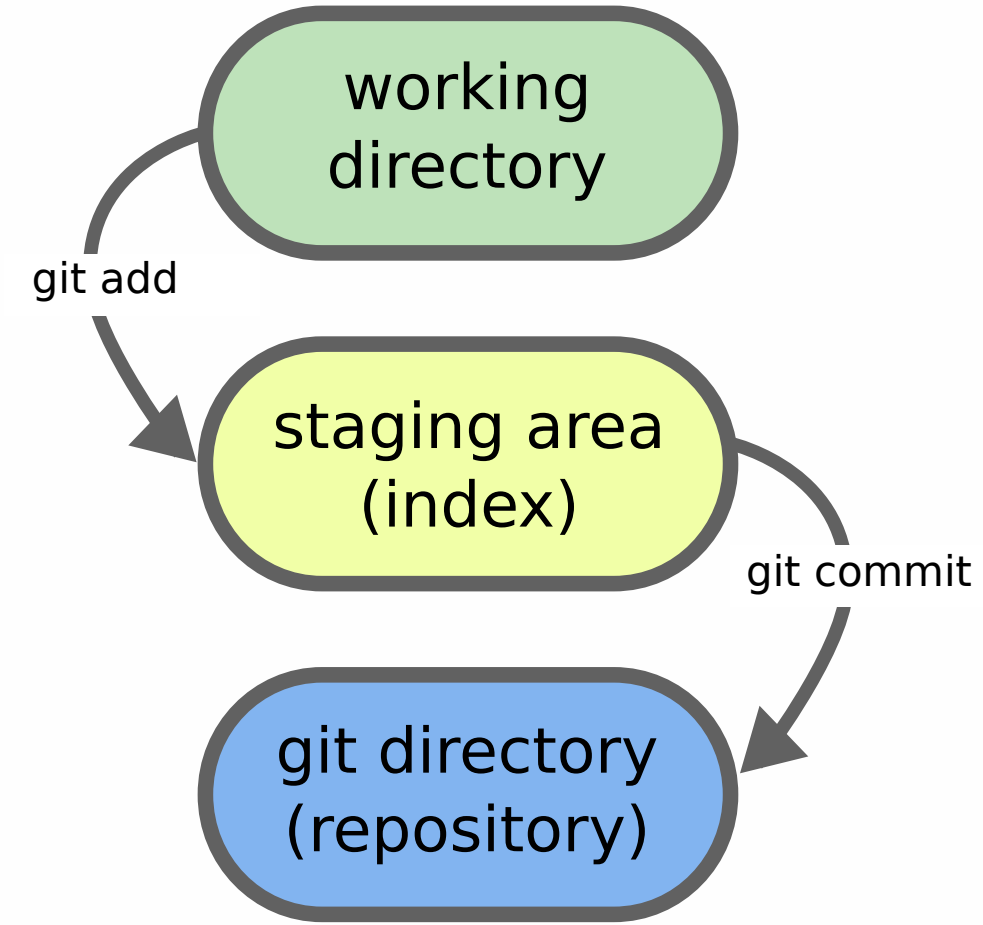
Os três passos

```
$ vim hello_world.py

$ cat hello_world.py
print "Hell World!"

$ git status

# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   hello_world.py
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Os três passos

Você **tem** que adicionar o arquivo ao *index* **depois** de editá-lo e/ou criá-lo.

Os três passos



Os três passos

```
$ git add hello_world.py
```

Os três passos

```
$ git add hello_world.py

$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   hello_world.py
#
```

Os três passos

```
$ git add hello_world.py

$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   hello_world.py
#

$ git commit -m "Hello World em Python"
```

Os três passos

```
$ git add hello_world.py

$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   hello_world.py
#

$ git commit -m "Hello World em Python"

[master e3d5175] Hello World em Python
 1 files changed, 1 insertions(+), 0 deletions(-)
```

Os três passos

```
$ git add hello_world.py

$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   hello_world.py
#

$ git commit -m "Hello World em Python"

[master e3d5175] Hello World em Python
 1 files changed, 1 insertions(+), 0 deletions(-)

$ git status
# On branch master
nothing to commit (working directory clean)
```

Os três passos

```
$ git log
commit e3d5175ca59febe510b1e0689040d2702b08c7ee
Author: Felipe Oliveira Carvalho <felipekde@gmail.com>
Date:   Sun Oct 2 17:36:41 2011 -0300

    Hello World em Python

commit 9b302737d41712809ca455b1d522c334793ef001
Author: Felipe Oliveira Carvalho <felipekde@gmail.com>
Date:   Sun Oct 2 15:45:23 2011 -0300

    Primeiro commit
```

Comandos vistos até agora

`git config` **Utilitário de configuração**

Comandos vistos até agora

`git config`	**Utilitário de configuração**
`git init`	**Cria um novo repositório**
`git clone`	**Clona um repositório existente**

Comandos vistos até agora

`git config`	**Utilitário de configuração**
`git init`	**Cria um novo repositório**
`git clone`	**Clona um repositório existente**
`git status`	**Mostra o estado do diretório de trabalho e index**

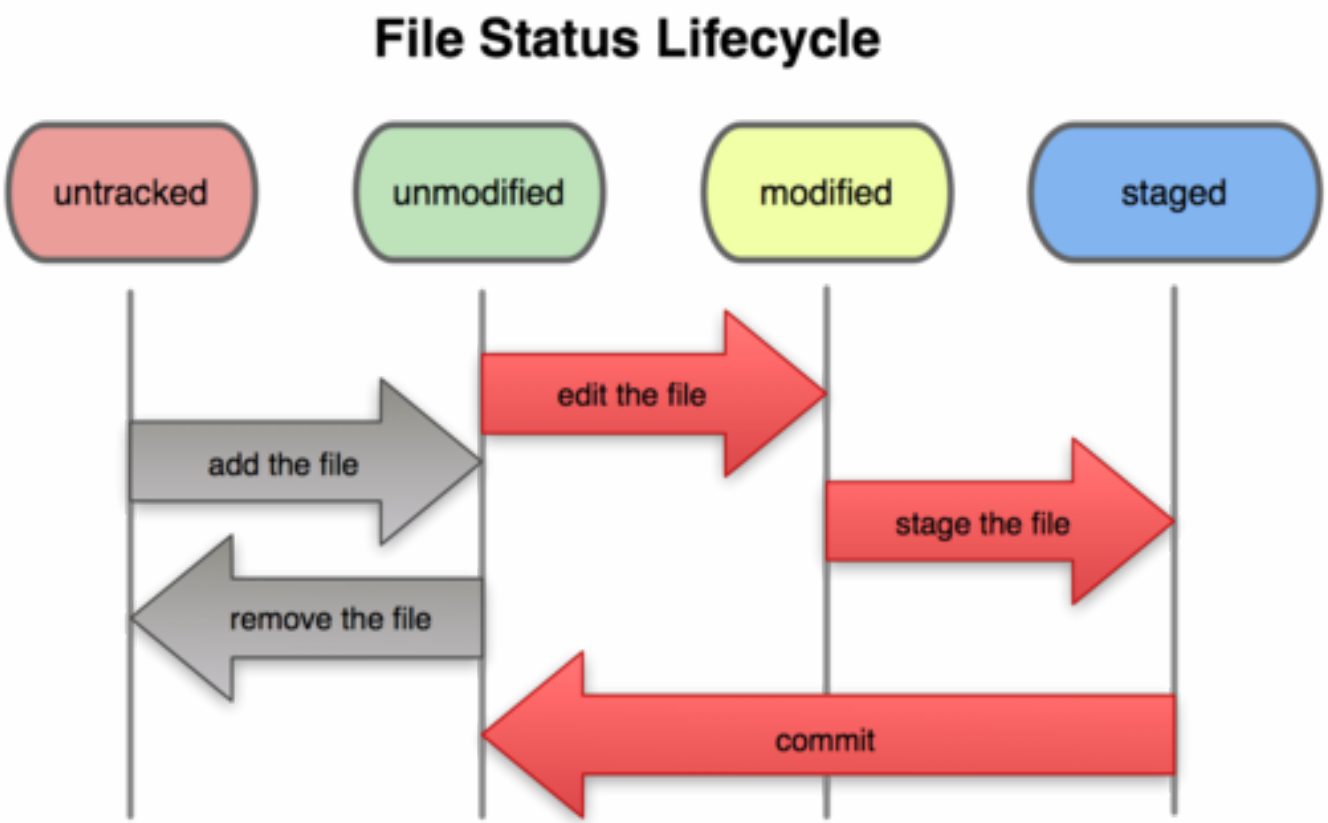
Comandos vistos até agora

`git config`	**Utilitário de configuração**
`git init`	**Cria um novo repositório**
`git clone`	**Clona um repositório existente**
`git status`	**Mostra o estado do diretório de trabalho e index**
`git add`	**Adiciona arquivos ao index**
`git commit`	**Faz o commit das mudanças no index**
`git show`	**Mostra detalhes do último commit**

Comandos vistos até agora

`git config`	**Utilitário de configuração**
`git init`	**Cria um novo repositório**
`git clone`	**Clona um repositório existente**
`git status`	**Mostra o estado do diretório de trabalho e index**
`git add`	**Adiciona arquivos ao index**
`git commit`	**Faz o commit das mudanças no index**
`git show`	**Mostra detalhes do último commit**
`git log`	**Lista os commits**

Untracked, Unmodified, Modified e Staged



Untracked, Unmodified, Modified e Staged

```
$ git status
# On branch master
nothing to commit (working directory clean)
```

Untracked, Unmodified, Modified e Staged

```
$ git status
# On branch master
nothing to commit (working directory clean)

$ vim README
```

Untracked, Unmodified, Modified e Staged

```
$ git status
# On branch master
nothing to commit (working directory clean)
```

```
$ vim README
```

Untracked

```
$ git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be co
#
#       README
nothing added to commit but untracked files present (use "g
```

Untracked, Unmodified, Modified e Staged

```
$ git add README
```

Untracked, Unmodified, Modified e Staged

```
$ git add README
```

Staged

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   README
#
```

Untracked, Unmodified, Modified e Staged

```
$ vim hello_git.py
```

Untracked, Unmodified, Modified e Staged

```
$ vim hello_git.py
```

Staged, Modified

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   README
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   hello_world.py
#
```

Untracked, Unmodified, Modified e Staged

```
$ git add hello_world.py
```

Untracked, Unmodified, Modified e Staged

```
$ git add hello_world.py
```

Staged

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   README
#       modified:   hello_world.py
#
```

Untracked, Unmodified, Modified e Staged

```
$ git commit -m "README e mudanças no Hello World"
[master 2b32cb9] README e mudanças no Hello World
 1 files changed, 1 insertions(+), 1 deletions(-)
 create mode 100644 README
```

Untracked, Unmodified, Modified e Staged

```
$ git commit -m "README e mudanças no Hello World"
[master 2b32cb9] README e mudanças no Hello World
 1 files changed, 1 insertions(+), 1 deletions(-)
 create mode 100644 README
```

Unmodified

```
$ git status
# On branch master
nothing to commit (working directory clean)
```

Visualizando modificações

```
$ vim hello_world.py
```

Visualizando modificações

```
$ vim hello_world.py

$ git status
[...]
$ git diff
[...]
$ git add hello_world.py
$ git status
[...]
$ git diff --staged
[...]
```

Removendo arquivos

```
$ rm README
```

Removendo arquivos

```
$ rm README

$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be comm
#   (use "git checkout -- <file>..." to discard changes in
#
#       deleted:    README
#
no changes added to commit (use "git add" and/or "git commi
```

Removendo arquivos

```
$ git rm README
rm 'README'
```

Removendo arquivos

```
$ git rm README
rm 'README'

$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       deleted:    README
#
```

Removendo arquivos

```
$ git rm README
rm 'README'

$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       deleted:    README
#

$ git commit -m "Removi o README"
[master b09a767] Removi o README
 0 files changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 README
```

Obtendo ajuda

```
$ git help <verb>
$ git <verb> --help
$ man git-<verb>
$ git <verb> -h
```

```
$ git commit -h
usage: git commit [options] [--] <filepattern>...

    -q, --quiet           suppress summary after successful commit
    -v, --verbose         show diff in commit message template

Commit message options
    -F, --file <file>     read message from file
    --author <author>     override author for commit
    --date <date>         override date for commit
    -m, --message <message>
                          commit message
    -c, --reedit-message <commit>
                          reuse and edit message from specified commit
    -C, --reuse-message <commit>
                          reuse message from specified commit
    --fixup <commit>      use autosquash formatted message to fixup specified commit
    --squash <commit>     use autosquash formatted message to squash specified commit
    --reset-author        the commit is authored by me now (used with -C-c/--amend)
    -s, --signoff         add Signed-off-by:
    -t, --template <file>
```

git log

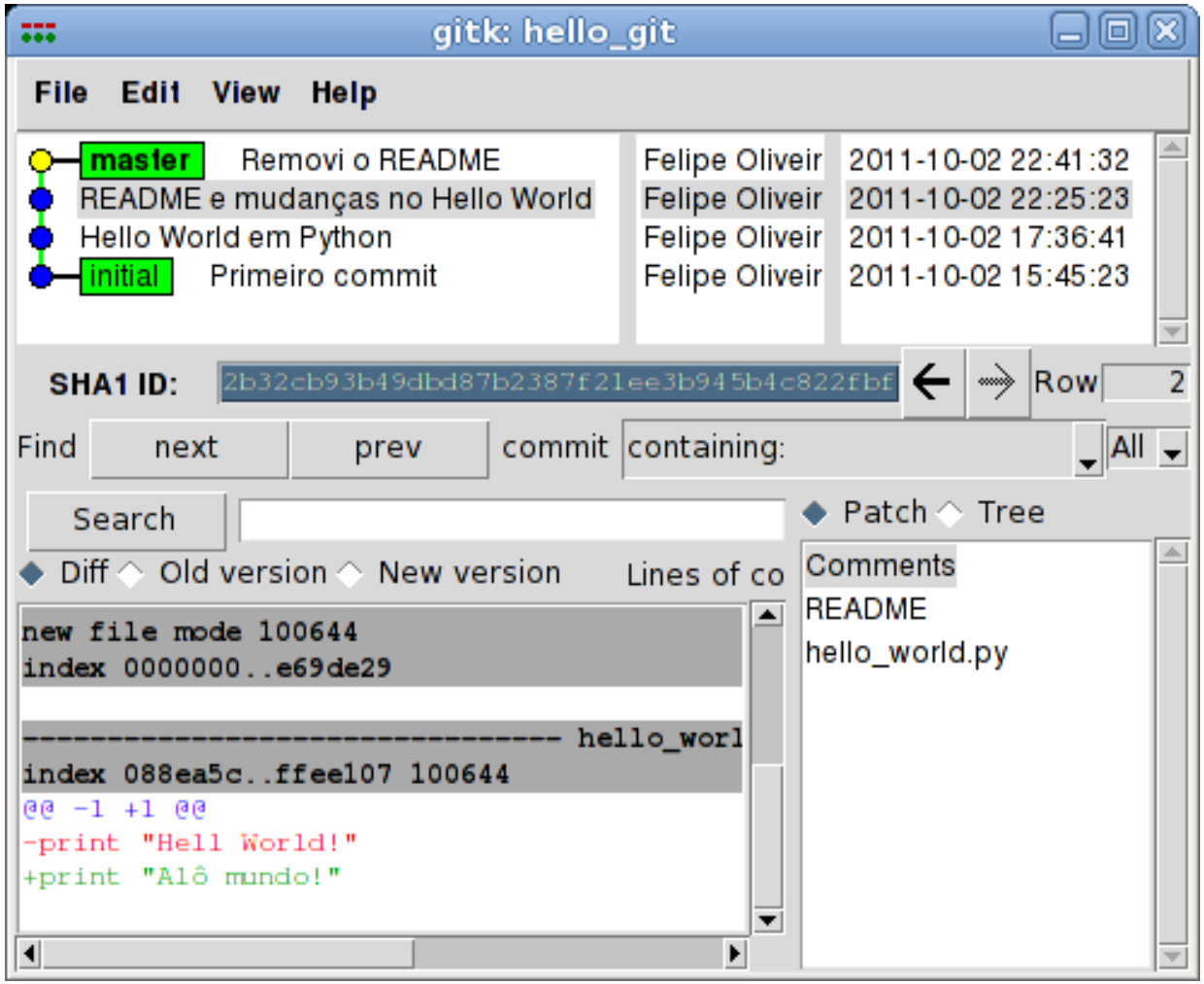
```
$ git log
commit b09a76779af9257ccd69c1fc5aac3e1dd0d03693
Author: Felipe Oliveira Carvalho <felipekde@gmail.com>
Date:   Sun Oct 2 22:41:32 2011 -0300

    Removi o README

commit 2b32cb93b49dbd87b2387f21ee3b945b4c822fbf
Author: Felipe Oliveira Carvalho <felipekde@gmail.com>
Date:   Sun Oct 2 22:25:23 2011 -0300

    README e mudanças no Hello World

commit e3d5175ca59febe510b1e0689040d2702b08c7ee
Author: Felipe Oliveira Carvalho <felipekde@gmail.com>
Date:   Sun Oct 2 17:36:41 2011 -0300

    Hello World em Python

commit 9b302737d41712809ca455b1d522c334793ef001
Author: Felipe Oliveira Carvalho <felipekde@gmail.com>
Date:   Sun Oct 2 15:45:23 2011 -0300

    Primeiro commit
```

Visualizando um commit

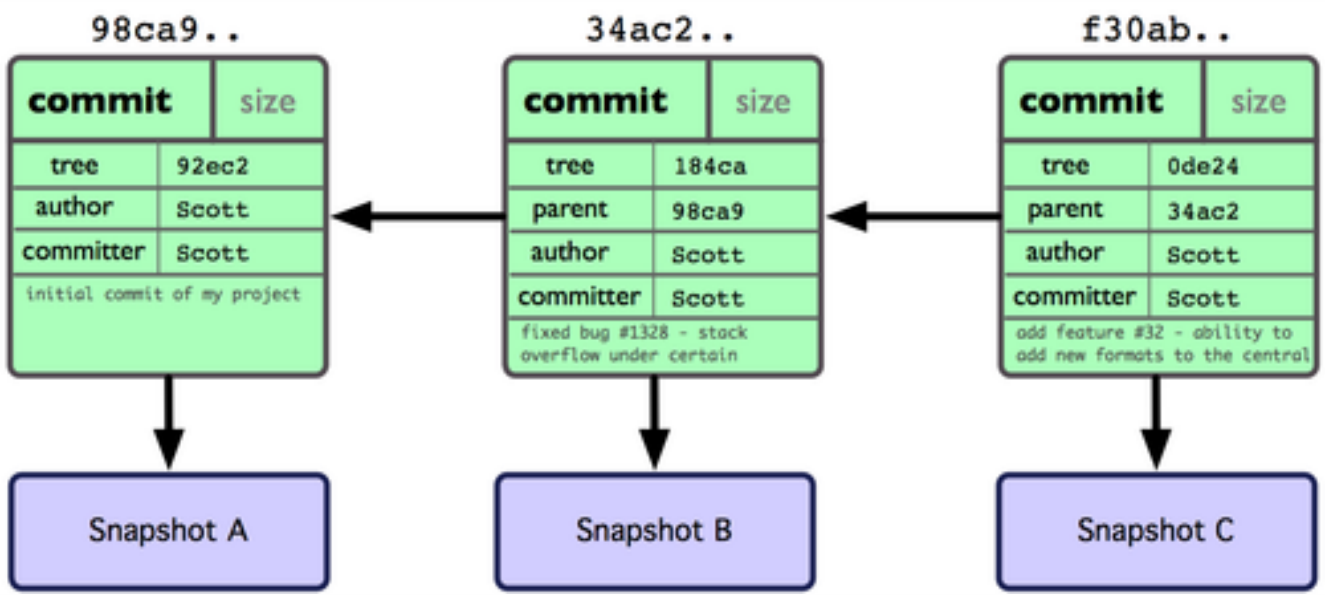
```
$ git show 2b32cb93
commit 2b32cb93b49dbd87b2387f21ee3b945b4c822fbf
Author: Felipe Oliveira Carvalho <felipekde@gmail.com>
Date:   Sun Oct 2 22:25:23 2011 -0300

    README e mudanças no Hello World

diff --git a/README b/README
new file mode 100644
index 0000000..e69de29
diff --git a/hello_world.py b/hello_world.py
index 088ea5c..ffee107 100644
--- a/hello_world.py
+++ b/hello_world.py
@@ -1 +1 @@
```

gitk

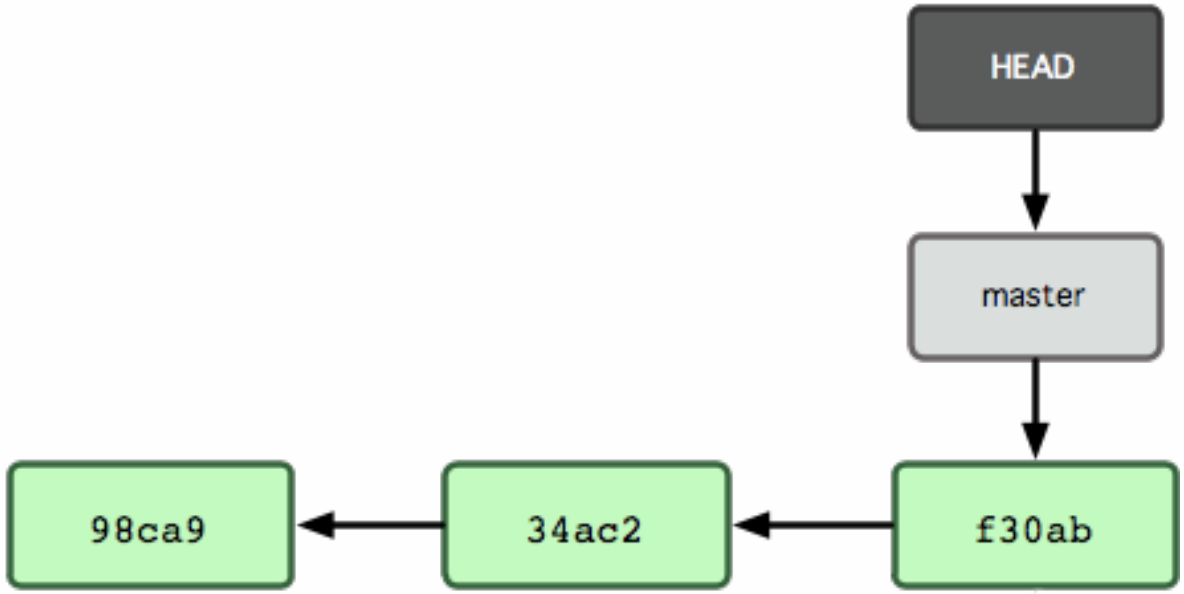
```
$ gitk &
```


gitk: hello_git
File Edit View Help
master Removi o README
README e mudanças no Hello World
Hello World em Python
initial Primeiro commit
Felipe Oliveir 2011-10-02 22:41:32
Felipe Oliveir 2011-10-02 22:25:23
Felipe Oliveir 2011-10-02 17:36:41
Felipe Oliveir 2011-10-02 15:45:23
SHA1 ID: 2b32cb93b49dbd87b2387f21ee3b945b4c822fbf
Row 2
Find next prev commit containing: All
Search
Patch Tree
Diff Old version New version Lines of co
Comments
README
hello_world.py
new file mode 100644
index 0000000..e69de29
--- hello_worl
index 088ea5c..ffee107 100644
@@ -1 +1 @@
-print "Hell World!"
+print "Alô mundo!"

Armazenamento de commits

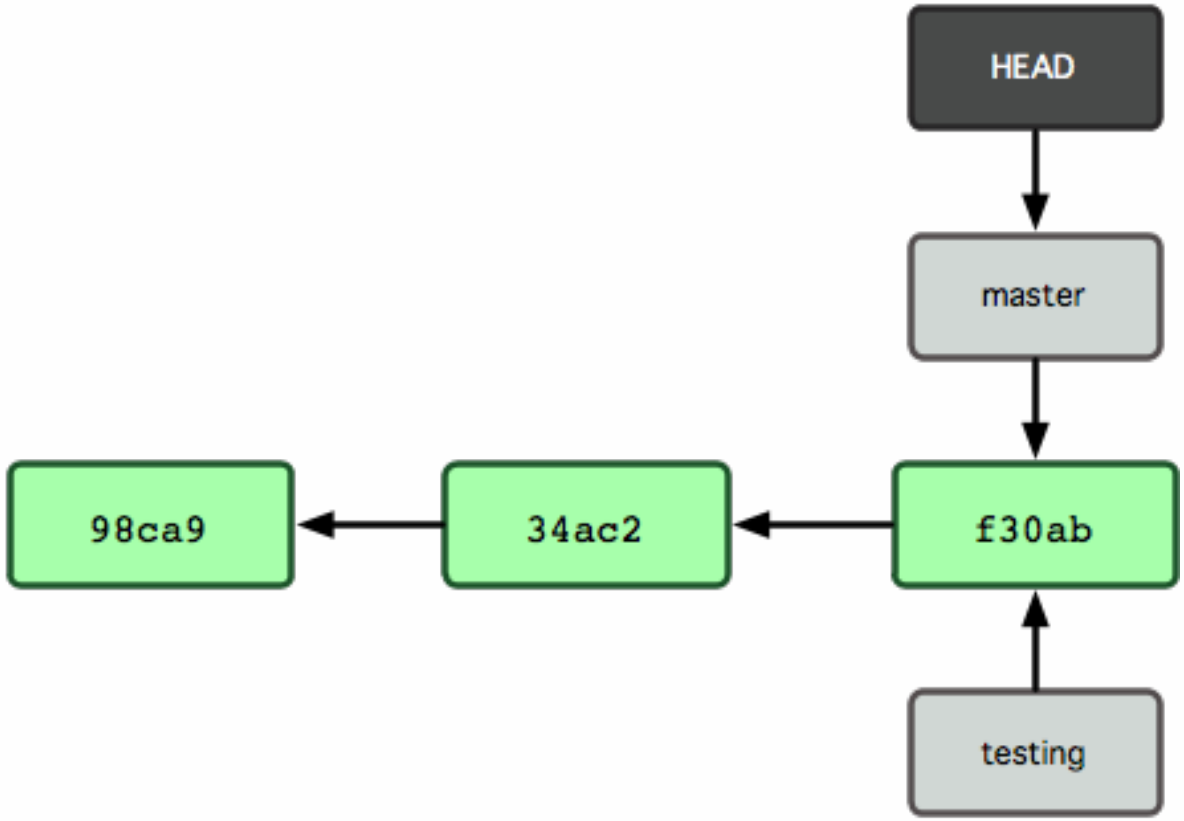


Branches



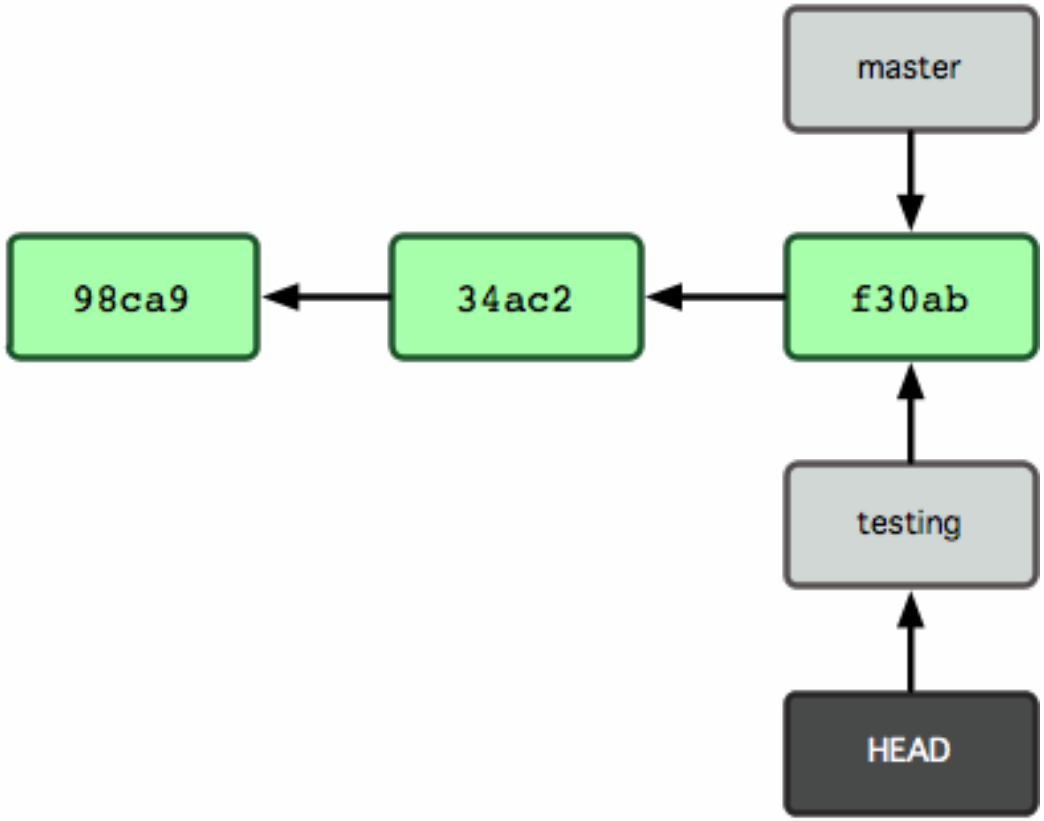
Criando um novo branch – branching

```
$ git branch testing
```



Selecionando um branch

```
$ git checkout testing
```

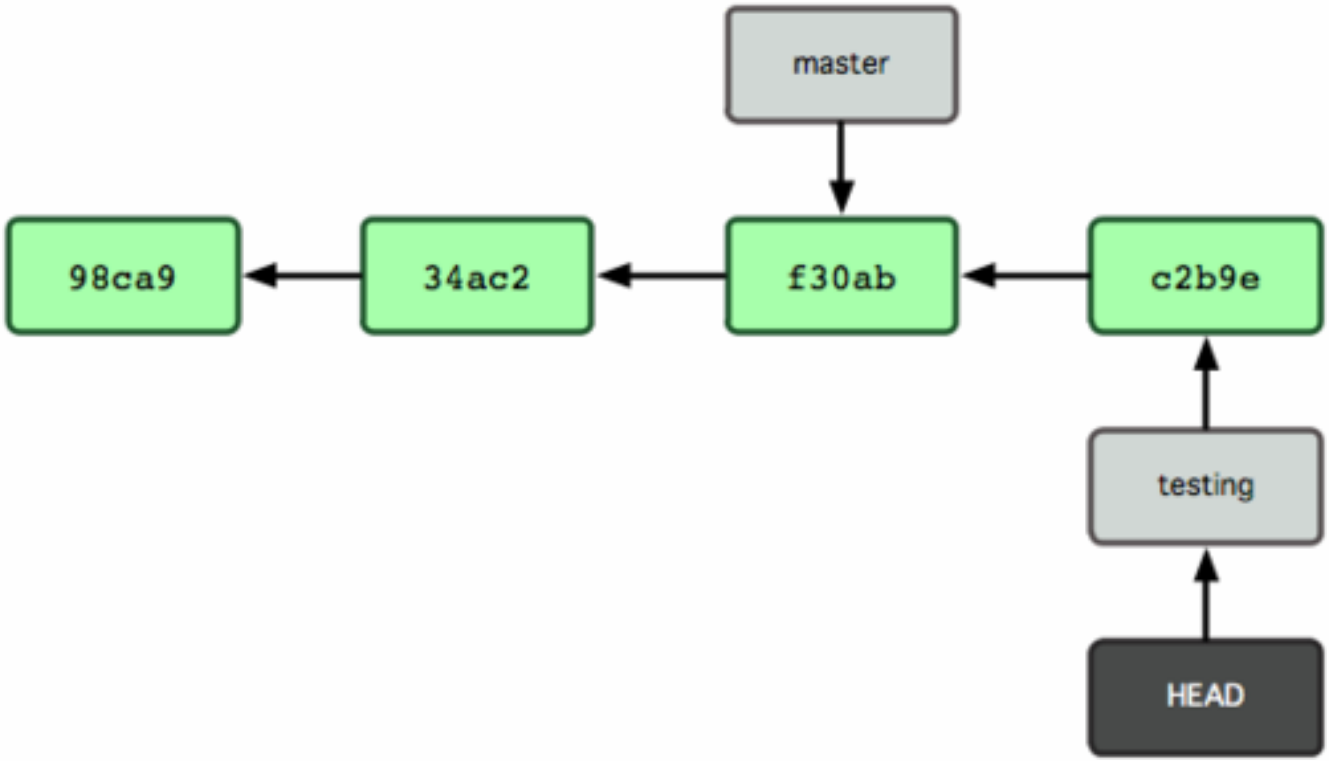


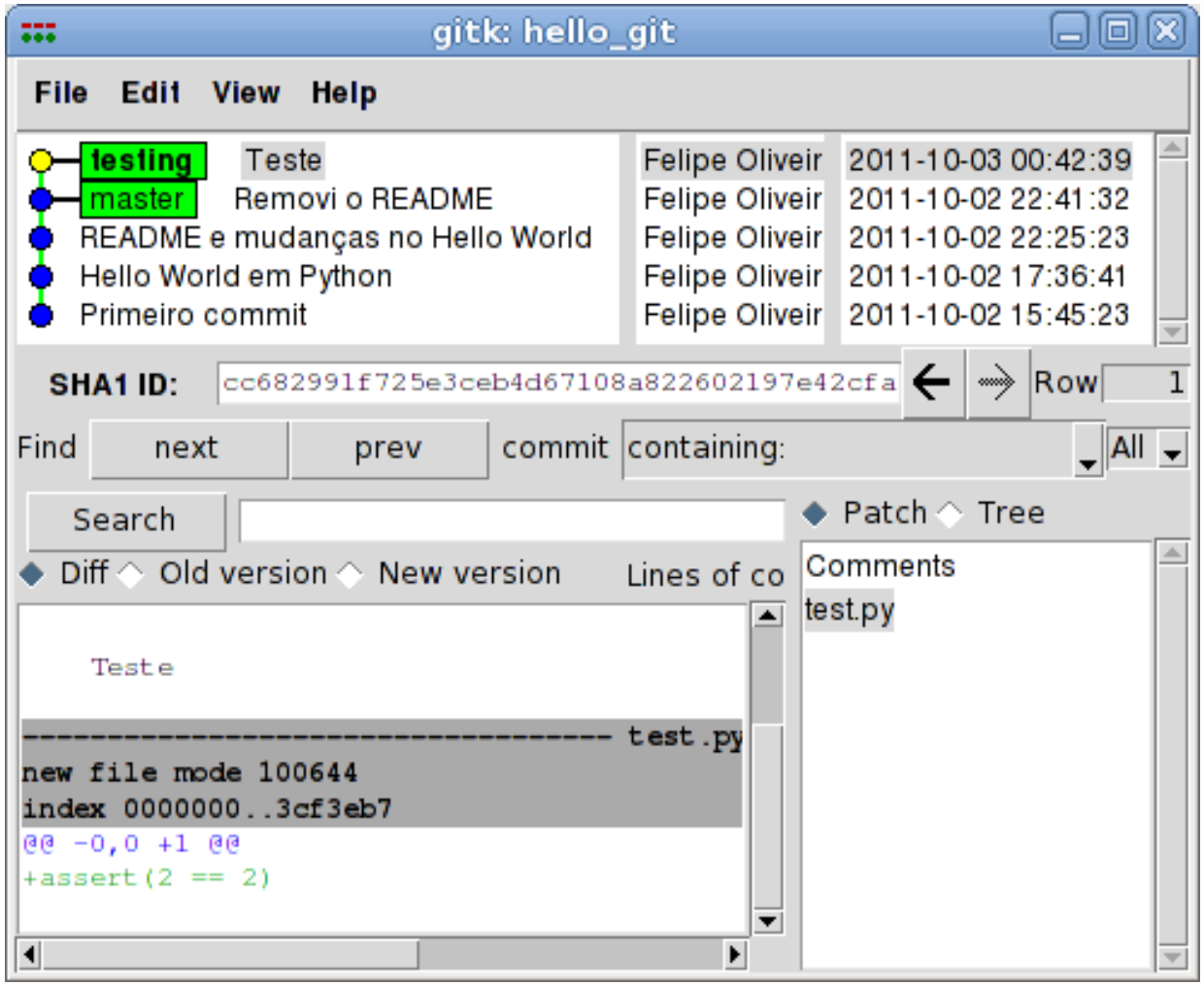
branch + checkout em um único comando

```
$ git checkout -b testing
Switched to a new branch 'testing'
```

Editando o novo branch

```
$ vim test.py
$ git add test.py
$ git commit -m "Teste"
```




gitk: hello_git
File Edit View Help
testing Teste
master Removi o README
README e mudanças no Hello World
Hello World em Python
Primeiro commit
Felipe Oliveir 2011-10-03 00:42:39
Felipe Oliveir 2011-10-02 22:41:32
Felipe Oliveir 2011-10-02 22:25:23
Felipe Oliveir 2011-10-02 17:36:41
Felipe Oliveir 2011-10-02 15:45:23
SHA1 ID: cc682991f725e3ceb4d67108a822602197e42cfa
Row 1
Find next prev commit containing: All
Search
Patch Tree
Diff Old version New version Lines of co
Comments
test.py
Teste
---------- test.py
new file mode 100644
index 0000000..3cf3eb7
@@ -0,0 +1 @@
+assert(2 == 2)

Merge simples – Fast-forward

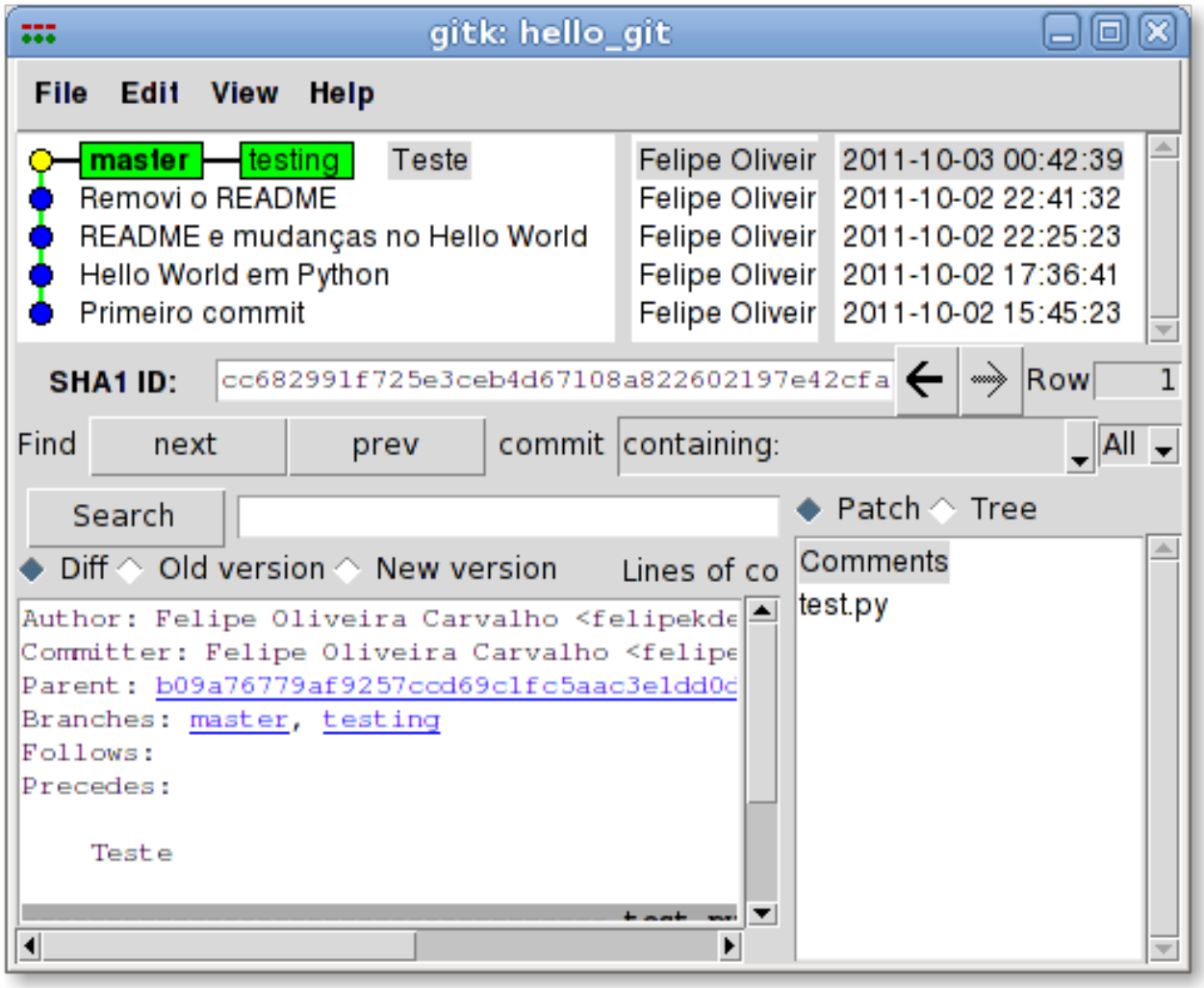
```
$ git checkout master
Switched to branch 'master'
```

Merge simples – Fast-forward

```
$ git checkout master
Switched to branch 'master'

$ git branch
* master
  testing
```

Merge simples – Fast-forward

```
$ git checkout master
Switched to branch 'master'

$ git branch
* master
  testing

$ git merge testing
Updating b09a767..cc68299
Fast-forward
 test.py |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 test.py
```


gitk: hello_git
File Edit View Help
master testing Teste
Removi o README
README e mudanças no Hello World
Hello World em Python
Primeiro commit
Felipe Oliveir 2011-10-03 00:42:39
Felipe Oliveir 2011-10-02 22:41:32
Felipe Oliveir 2011-10-02 22:25:23
Felipe Oliveir 2011-10-02 17:36:41
Felipe Oliveir 2011-10-02 15:45:23
SHA1 ID: cc682991f725e3ceb4d67108a822602197e42cfa
Row 1
Find next prev commit containing: All
Search
Patch Tree
Diff Old version New version Lines of co
Comments
test.py
Author: Felipe Oliveira Carvalho <felipekde
Committer: Felipe Oliveira Carvalho <felipe
Parent: b09a76779af9257ccd69c1fc5aac3e1dd0c
Branches: master, testing
Follows:
Precedes:
Teste

Um exemplo

- Clone o código que está na produção
- Crie um branch para issue #53 (‘iss53’)
- Trabalhe por 10 minutos
- Alguém pede um hotfix para a issue #102
- checkout ‘master’
- Crie o branch ‘iss102’
- Resolva o problema
- checkout ‘master‘, merge ‘iss102’
- push para a versão pública
- checkout ‘iss53’ e continue trabalhando

Mais situações

Isolar unidades de trabalho

Mais situações

Você quer **experimentar** alguma ideia

Mais situações

Você vai fazer algo que vai **demorar**

Passo-a-passo do exemplo

Resolva a issue 53. Crie um branch para isso a partir do `master`:

Passo-a-passo do exemplo

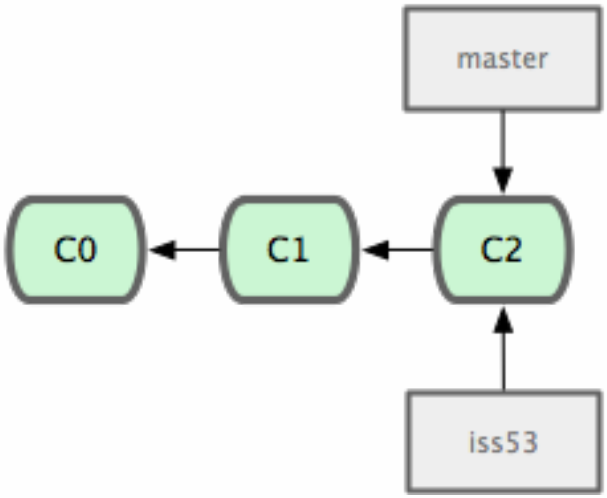
Resolva a issue 53. Crie um branch para isso a partir do `master`:

```
$ git checkout -b iss53
Switched to a new branch 'iss53'
```

Passo-a-passo do exemplo

Resolva a issue 53. Crie um branch para isso a partir do `master`:

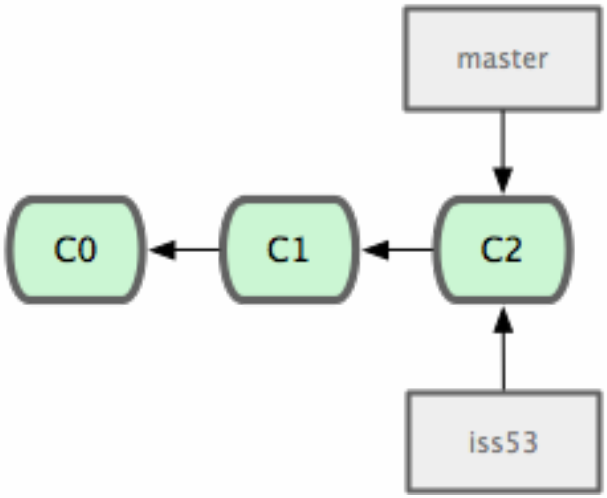
```
$ git checkout -b iss53
Switched to a new branch 'iss53'
```



Passo-a-passo do exemplo

Resolva a issue 53. Crie um branch para isso a partir do `master`:

```
$ git checkout -b iss53
Switched to a new branch 'iss53'
```



Corrija o problema no código:

```
$ vim hello_world.py
```

Passo-a-passo do exemplo

```
$ git diff
diff --git a/hello_world.py b/hello_world.py
index ffee107..349aa41 100644
--- a/hello_world.py
+++ b/hello_world.py
@@ -1 +1 @@
-print "Alô mundo!"
+print "Alô, mundo!"
```

Passo-a-passo do exemplo

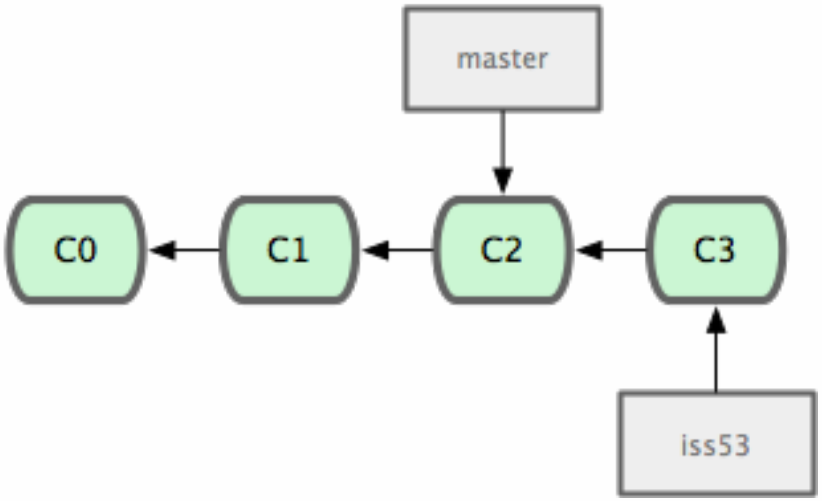
commit das mudanças feitas:

```
$ git commit -a -m "Vírgula adicionada [iss53]"
[iss53 63f9671] Vírgula adicionada [iss53]
 1 files changed, 1 insertions(+), 1 deletions(-)
```

Passo-a-passo do exemplo

commit das mudanças feitas:

```
$ git commit -a -m "Vírgula adicionada [iss53]"
[iss53 63f9671] Vírgula adicionada [iss53]
 1 files changed, 1 insertions(+), 1 deletions(-)
```



Passo-a-passo do exemplo

Uma feature tem que ser implementada agora! Faça `checkout` do `master`, pois a nova feature vai ser implementada a partir do código estável: supostamente o código no *branch* `master`.

```
$ git checkout master
Switched to branch 'master'
```

Passo-a-passo do exemplo

Uma feature tem que ser implementada agora! Faça `checkout` do `master`, pois a nova feature vai ser implementada a partir do código estável: supostamente o código no *branch* `master`.

```
$ git checkout master
Switched to branch 'master'
```

Crie um novo *branch* – `bomdia` – para implementar a nova feature:

```
$ git checkout -b bomdia
Switched to a new branch 'bomdia'
```

Passo-a-passo do exemplo

Uma feature tem que ser implementada agora! Faça `checkout` do `master`, pois a nova feature vai ser implementada a partir do código estável: supostamente o código no *branch* `master`.

```
$ git checkout master
Switched to branch 'master'
```

Crie um novo *branch* – `bomdia` – para implementar a nova feature:

```
$ git checkout -b bomdia
Switched to a new branch 'bomdia'
```

```
$ vim bom_dia.py
```

Passo-a-passo do exemplo

```
$ git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be co
#
#       bom_dia.py
nothing added to commit but untracked files present (use "g
```

Passo-a-passo do exemplo

```
$ git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be co
#
#       bom_dia.py
nothing added to commit but untracked files present (use "g

$ git add bom_dia.py
```

Passo-a-passo do exemplo

```
$ git diff --staged
diff --git a/bom_dia.py b/bom_dia.py
new file mode 100644
index 0000000..b62bb76
--- /dev/null
+++ b/bom_dia.py
@@ -0,0 +1 @@
+print "Bom dia!"
```

Passo-a-passo do exemplo

`commit` do "bom dia":

Passo-a-passo do exemplo

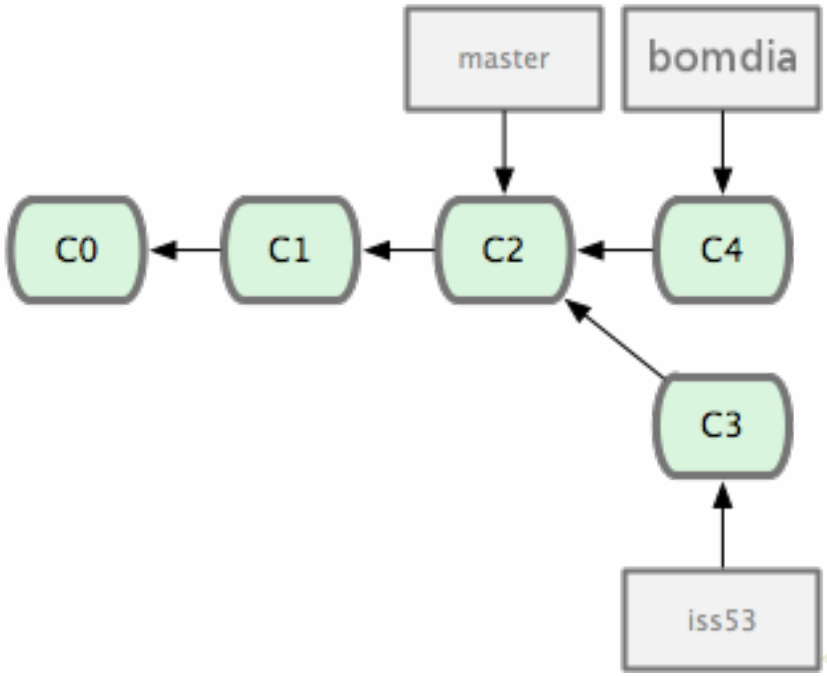
commit do "bom dia":

```
$ git commit -a # use o editor para inserir a mensagem
[master cf9146f] Nova feature: script que diz "bom dia"
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 bom_dia.py
```

Passo-a-passo do exemplo

commit do "bom dia":

```
$ git commit -a # use o editor para inserir a mensagem
[master cf9146f] Nova feature: script que diz "bom dia"
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 bom_dia.py
```



Passo-a-passo do exemplo

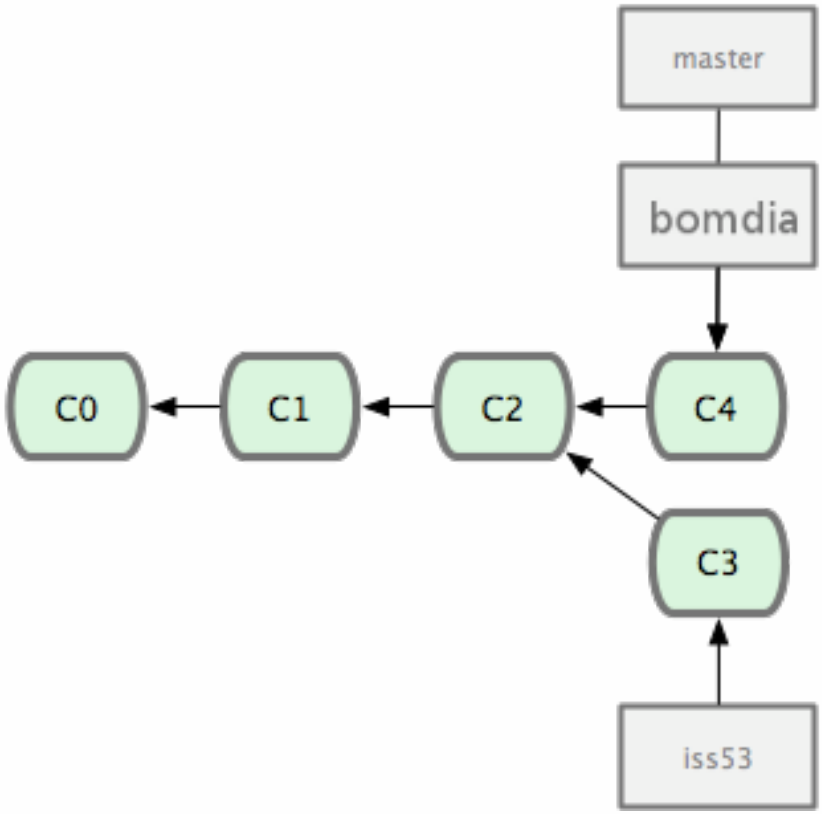
Adicionando mudanças ao branch *branch* `master`.

Passo-a-passo do exemplo

Adicionando mudanças ao branch *branch* `master`.

```
$ git checkout master
$ git merge bomdia
Updating cc68299..296d018
Fast-forward
 bom_dia.py |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 bom_dia.py
```

Passo-a-passo do exemplo



Passo-a-passo do exemplo

O `merge` já foi feito, então você pode deletar o *branch* `bomdia`. `master` contém o código com a feature que foi solicitada. Então esse código pode ser enviado para a produção (ou não).

Passo-a-passo do exemplo

O `merge` já foi feito, então você pode deletar o *branch* `bomdia`. `master` contém o código com a feature que foi solicitada. Então esse código pode ser enviado para a produção (ou não).

```
$ git branch -d bomdia
Deleted branch bomdia (was 296d018).
```

Passo-a-passo do exemplo

O `merge` já foi feito, então você pode deletar o *branch* `bomdia`. `master` contém o código com a feature que foi solicitada. Então esse código pode ser enviado para a produção (ou não).

```
$ git branch -d bomdia
Deleted branch bomdia (was 296d018).
```

E agora você pode continuar a trabalhar na resulução da issue #53.

Passo-a-passo do exemplo

O `merge` já foi feito, então você pode deletar o *branch* `bomdia`. `master` contém o código com a feature que foi solicitada. Então esse código pode ser enviado para a produção (ou não).

```
$ git branch -d bomdia
Deleted branch bomdia (was 296d018).
```

E agora você pode continuar a trabalhar na resulução da issue #53.

```
$ git checkout iss53
Switched to branch 'iss53'
```

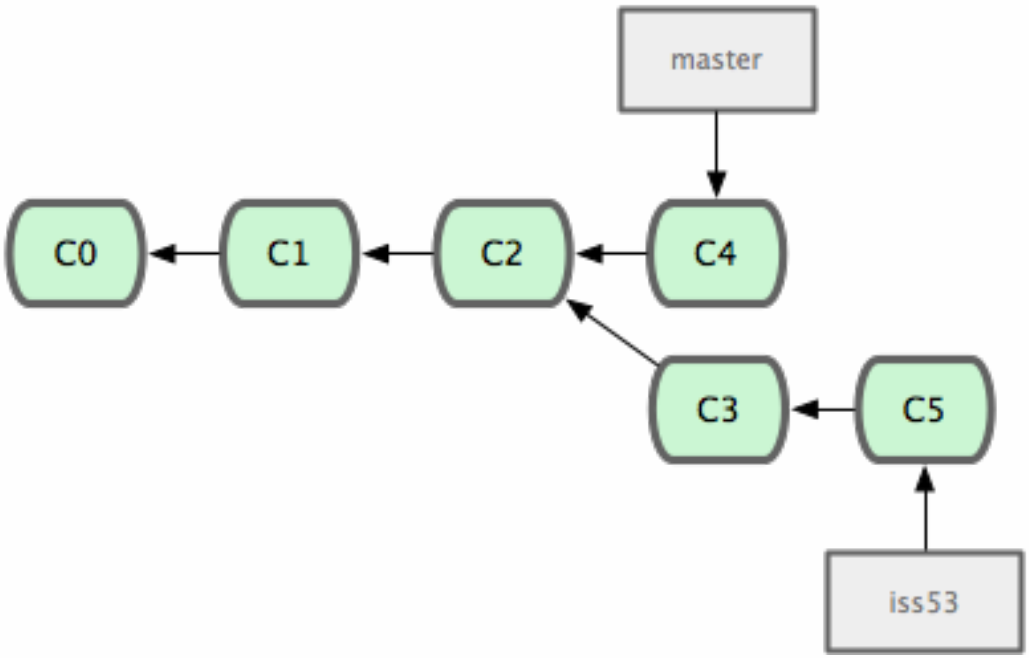
Passo-a-passo do exemplo

Passo-a-passo do exemplo

```
$ vim hello_world.py
$ git commit -a -m "mundo => Mundo"
[iss53 e4c9096] mundo => Mundo
 1 files changed, 1 insertions(+), 1 deletions(-)
```

Passo-a-passo do exemplo

```
$ vim hello_world.py
$ git commit -a -m "mundo => Mundo"
[iss53 e4c9096] mundo => Mundo
 1 files changed, 1 insertions(+), 1 deletions(-)
```

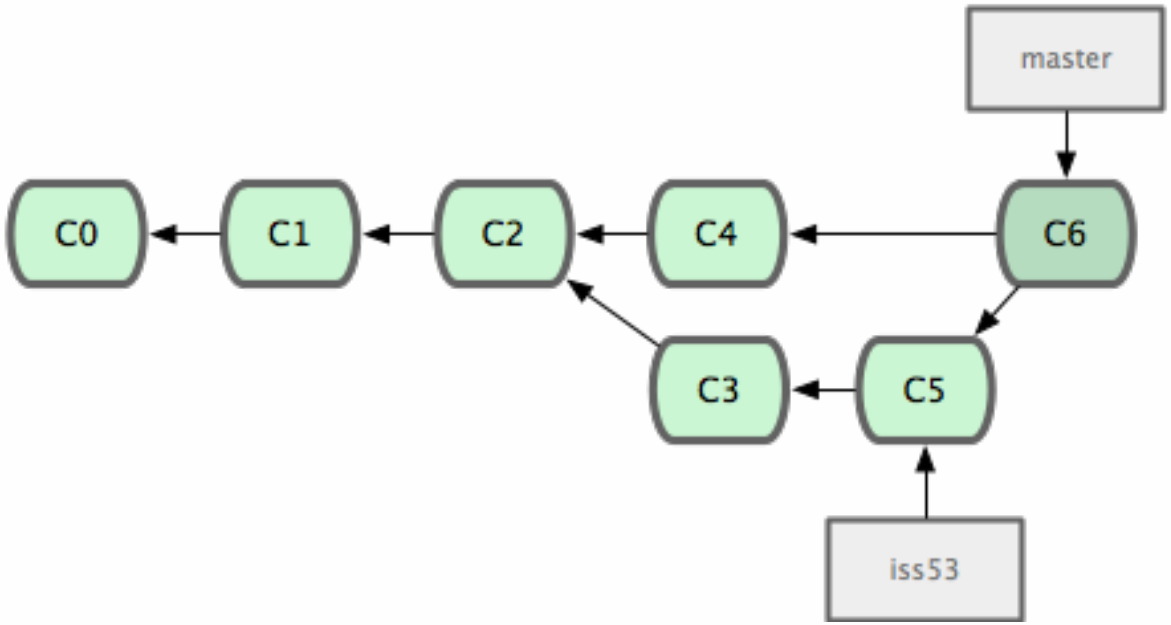


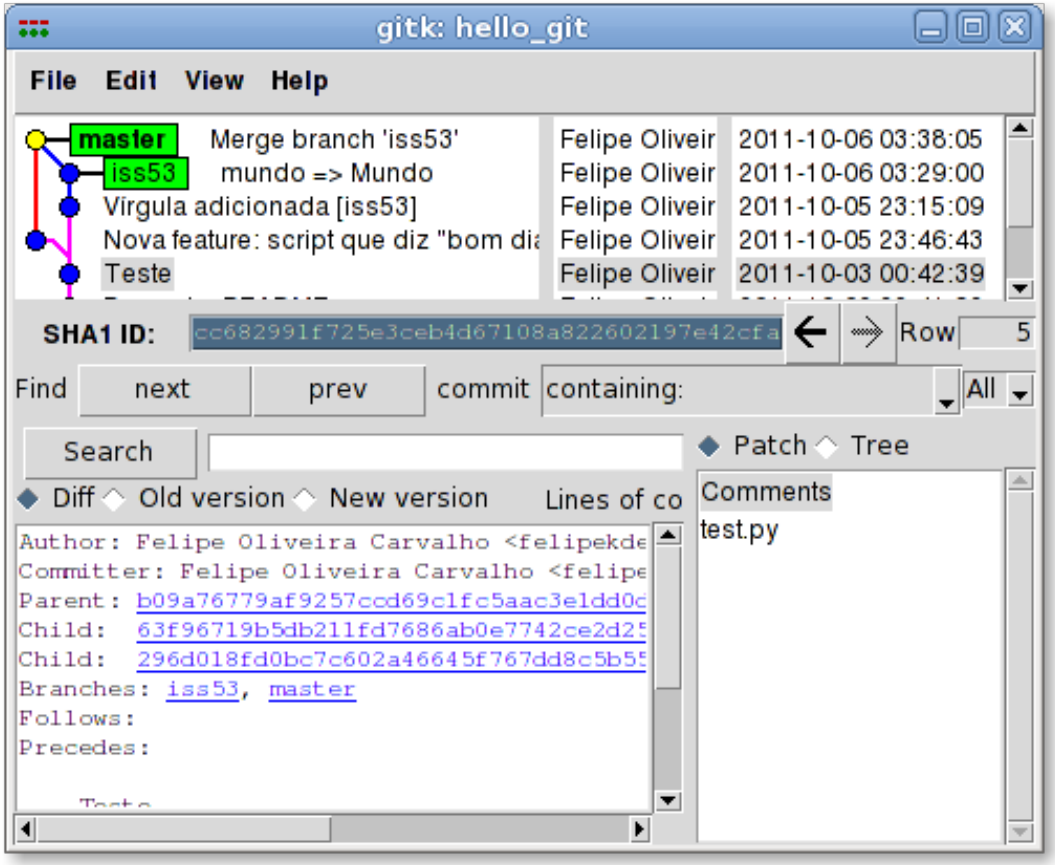
Passo-a-passo do exemplo

merge `iss53` com o `master`.

```
$ git checkout master
Switched to branch 'master'
$ git merge iss53
Merge made by recursive.
 hello_world.py |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)
```

Passo-a-passo do exemplo




gitk: hello_git
File Edit View Help
master
iss53
Merge branch 'iss53'
mundo => Mundo
Vírgula adicionada [iss53]
Nova feature: script que diz "bom dia
Teste
Felipe Oliveir
2011-10-06 03:38:05
2011-10-06 03:29:00
2011-10-05 23:15:09
2011-10-05 23:46:43
2011-10-03 00:42:39
SHA1 ID: cc682991f725e3ceb4d67108a822602197e42cfa
Row 5
Find next prev commit containing: All
Search
Patch Tree
Diff Old version New version Lines of co
Comments
test.py
Author: Felipe Oliveira Carvalho <felipekde
Committer: Felipe Oliveira Carvalho <felipe
Parent: b09a76779af9257ccd69c1fc5aac3eldd0c
Child: 63f96719b5db211fd7686ab0e7742ce2d25
Child: 296d018fd0bc7c602a46645f767dd8c5b55
Branches: iss53, master
Follows:
Precedes:

```
$ git merge umbranch
Auto-merging hello_world.py
CONFLICT (content): Merge conflict in hello_world.py
Automatic merge failed; fix conflicts and then commit the r
```

```
$ git merge umbranch
Auto-merging hello_world.py
CONFLICT (content): Merge conflict in hello_world.py
Automatic merge failed; fix conflicts and then commit the r

$ cat hello_world.py
<<<<<<< HEAD
print "Alô, Mundo!" # Imprime Alô, Mundo
=======
print "Alô, Mundo!" # LOL
>>>>>>> umbranch
```

Criando um projeto no servidor

```
$ git clone --bare my_project my_project.git
Initialized empty Git repository in /opt/projects/my_projec
```

Criando um projeto no servidor

```
$ git clone --bare my_project my_project.git
Initialized empty Git repository in /opt/projects/my_projec

$ scp -r my_project.git user@git.example.com:/opt/git
```

Criando um projeto no servidor

```
$ git clone --bare my_project my_project.git
Initialized empty Git repository in /opt/projects/my_projec
```

```
$ scp -r my_project.git user@git.example.com:/opt/git
```

Outro usuário clona o repositório

```
$ git clone user@git.example.com:/opt/git/my_project.git
```

Criando um projeto no servidor

```
$ git clone --bare my_project my_project.git
Initialized empty Git repository in /opt/projects/my_projec

$ scp -r my_project.git user@git.example.com:/opt/git
```

Outro usuário clona o repositório

```
$ git clone user@git.example.com:/opt/git/my_project.git

$ ssh user@git.example.com
$ cd /opt/git/my_project.git
$ git init --bare --shared
```

Outros usuários podem enviar as mudanças para o repositório no servidor.

```
$ vim README
$ git commit -am 'fix for the README file'
$ git push origin master
```

github

http://github.com

Workflow básico

Usuário clona o repositório remoto com `git clone` e trabalha localmente.

Workflow básico

Usuário clona o repositório remoto com `git clone` e trabalha localmente.

```
$ vim TODO
$ git commit -a
```

Workflow básico

Usuário clona o repositório remoto com `git clone` e trabalha localmente.

```
$ vim TODO
$ git commit -a
```

Usuário quer enviar suas mudanças para o servidor:

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

git pull

```
git pull = fetch + merge
```

Comandos úteis

- `git blame`

Comandos úteis

- `git blame`
- `git grep`

Comandos úteis

- `git blame`
- `git grep`
- `git cherry-pick`

Comandos úteis

- `git blame`
- `git grep`
- `git cherry-pick`
- `git stash`

git stash

save

git stash

save

list

git stash

save

list

drop

git stash

save

list

drop

pop

git stash

save

list

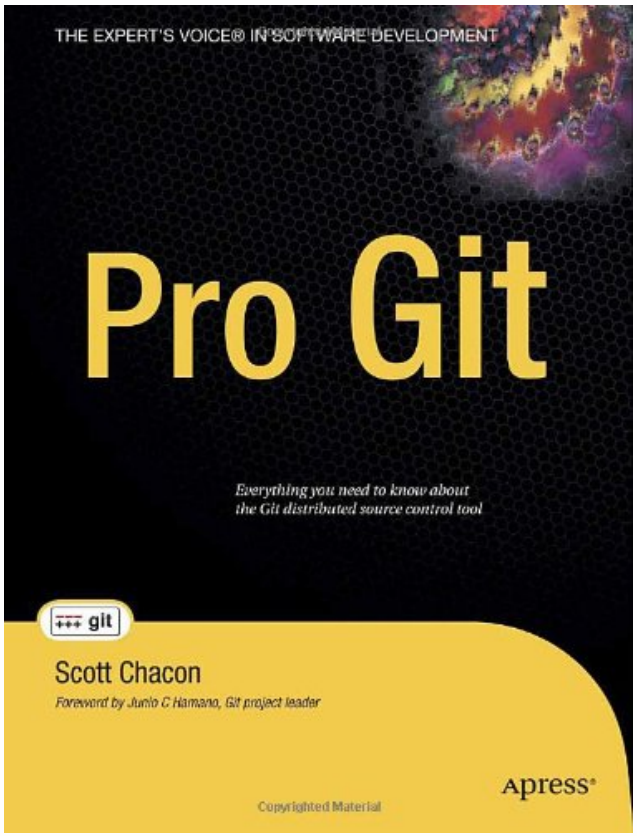
drop

pop

apply

Pro Git

http://progit.org



Mais perguntas?

Mais perguntas?