Git – Um Sistema de Controle de Versões Distribuído

Fernando Castor

Centro de Informática – Universidade Federal de Pernambuco

Índice

Um pouco de história

Quem é este?

Quem é este?

Linus B. Torvalds

Contexto

Algumas informações sobre o Linux

- 9516 *committers*
- Até 11000 *commits* por mês
- Quase 25 milhões de linhas de código
- Mais informações: http://www.ohloh.net/p/linux/

Contexto

Algumas informações sobre o Linux

- 9516 *committers*
- Até 11000 *commits* por mês
- Quase 25 milhões de linhas de código
- Mais informações: http://www.ohloh.net/p/linux/

Linux e SCVs

- Até 2002: nenhum (apenas *patches*)
- Entre 2002 e 2005: BitKeeper
  - Sistema comercial (mas gratuito)
- A partir de 2005: Git
  - Devido a problemas com o proprietário do BitKeeper

Objetivos de projeto do Git

Em resumo: Deveria ser o **SCV do Linux**

Objetivos de projeto do Git

Em resumo: Deveria ser o **SCV do Linux**

Mais especificamente

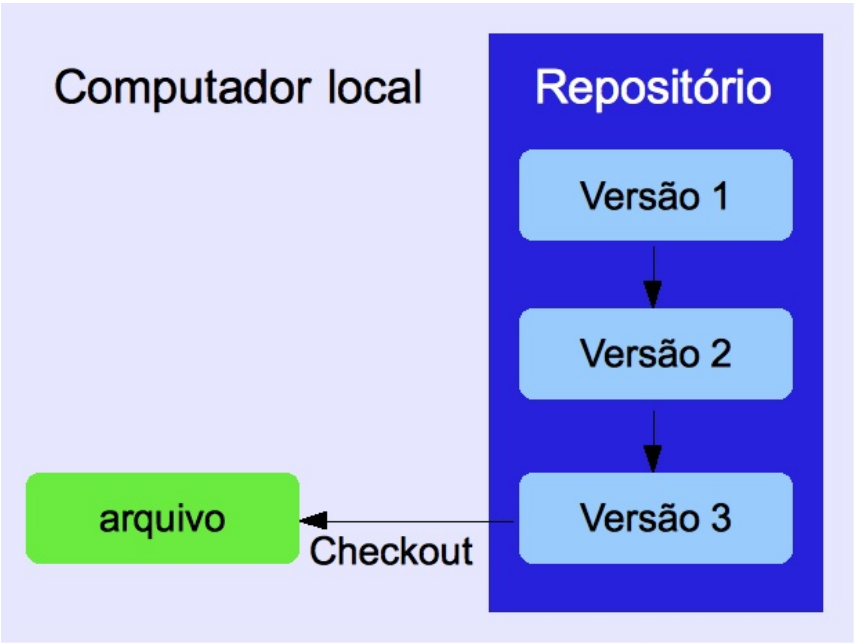
1. Ser diferente do CVS (e do SVN!)
2. Suporte a controle de versões distribuído
3. Salvaguardas contra corrupção de repositórios
4. **Excelente** desempenho

Git hoje

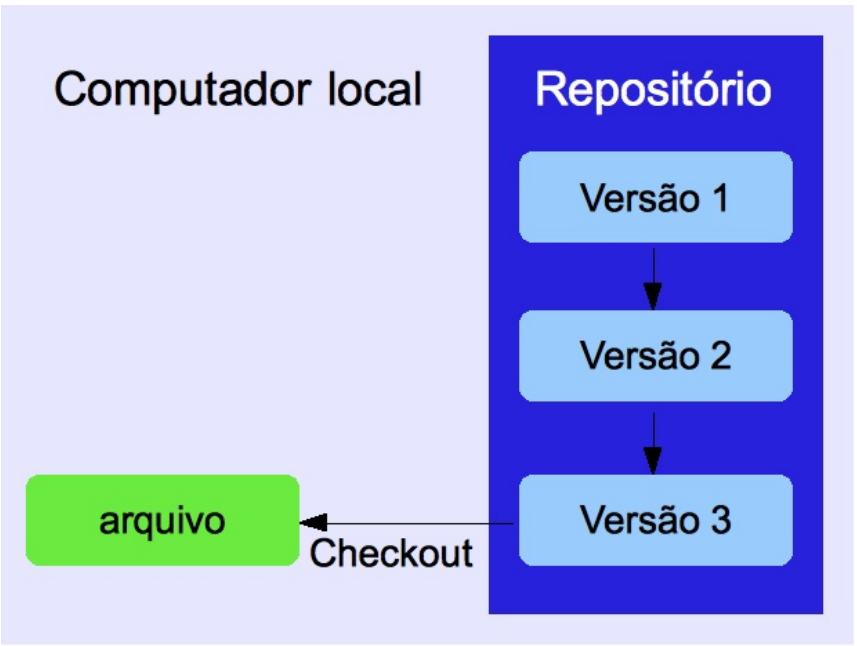
1. Mantido por Junio Hamano
2. Disponível para várias plataformas
3. Atualmente na versão 1.8.4
4. Já é um dos SCVs mais usados
5. Vários sítios de *hosting* dão suporte ao Git
   - Alguns lembram redes sociais

Controle de Versões Distribuídos

SCV local


Computador local
Repositório
Versão 1
Versão 2
Versão 3
arquivo
Checkout

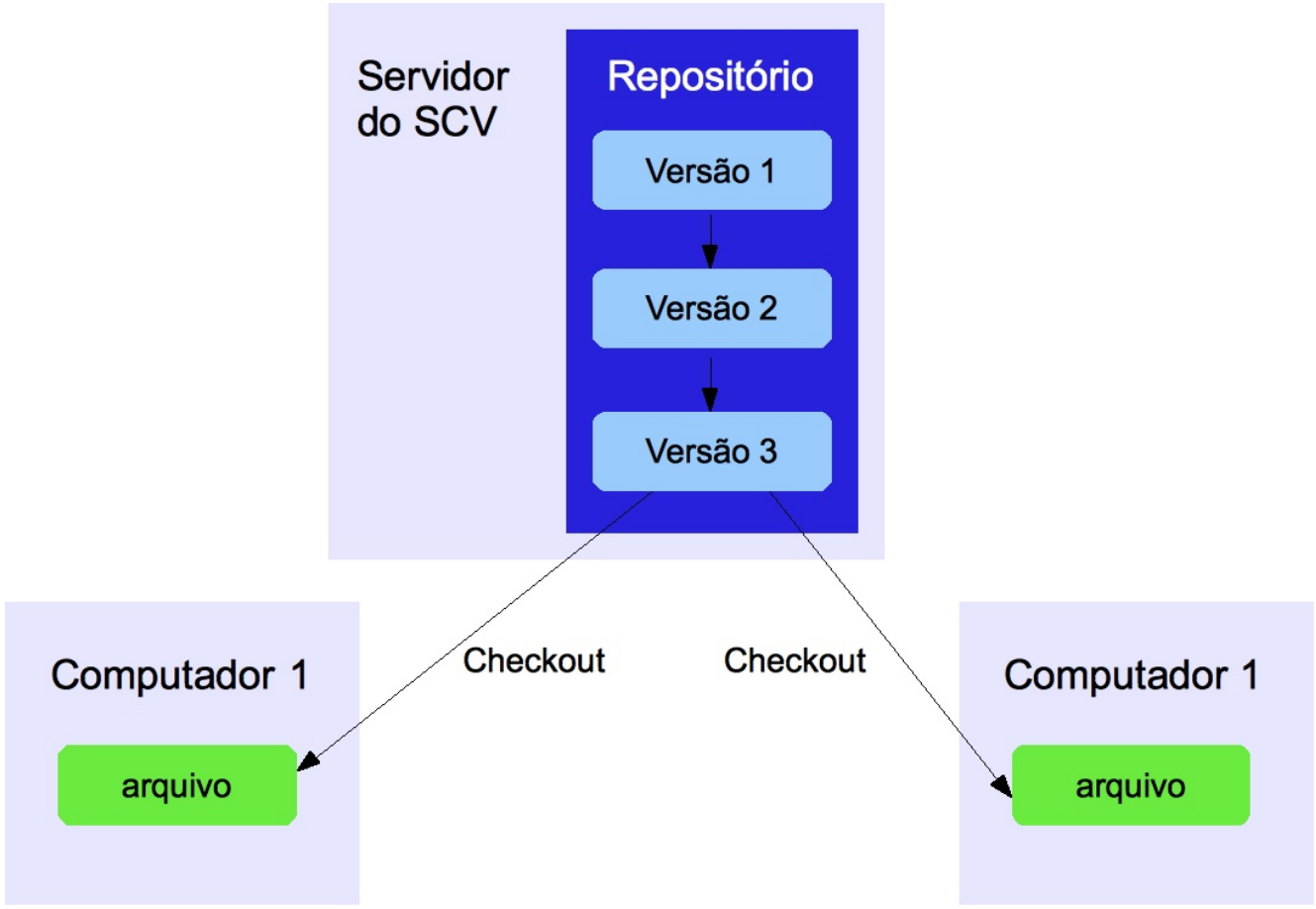
SCV local



Características de SCVs locais

- \+ Possibilidade de recriar qualquer versão
- \+ Forma organizada de fazer backups
- − Colaboração é difícil

SCV com repositório centralizado



SCV com repositório centralizado

Vantagens

- \+ Possibilidade de colaboração
- \+ Todos podem saber o que os outros fazem
- \+ Administradores têm controle sobre permissões

SCV com repositório centralizado

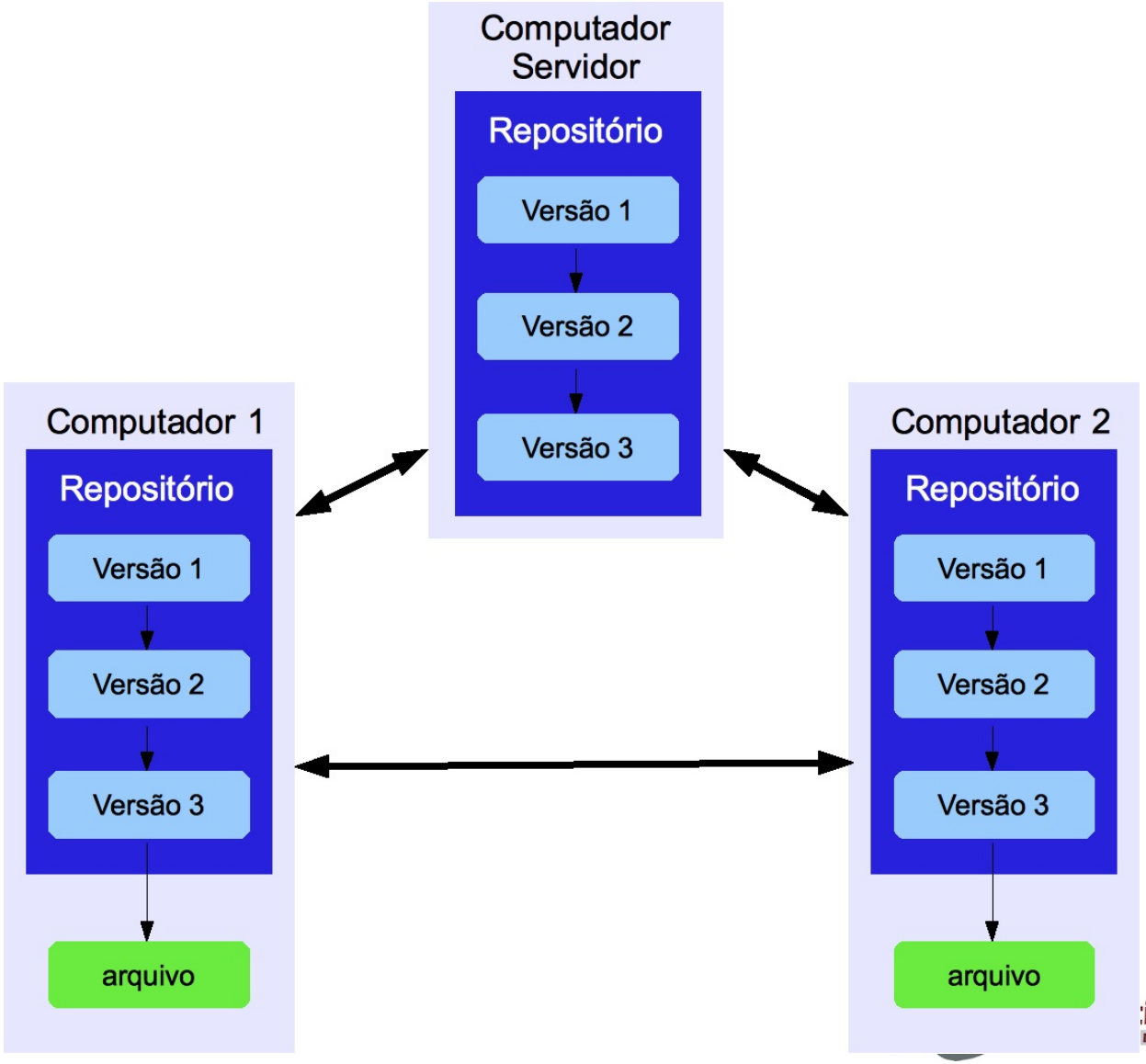
Vantagens

- \+ Possibilidade de colaboração
- \+ Todos podem saber o que os outros fazem
- \+ Administradores têm controle sobre permissões

Desvantagens

- − Ponto único de falha
- − Conexão com o servidor é necessária
- − Gargalo de desempenho

SCV distribuído



SCV distribuído

Vantagens

- \+ Todas as vantagens de um SCV local ou com repositório centralizado
- \+ Menos vulnerabilidade a corrupção de dados
- \+ Excelente desempenho
- \+ Possibilidades diversas de fluxos de trabalho

SCV distribuído

Vantagens

- \+ Todas as vantagens de um SCV local ou com repositório centralizado
- \+ Menos vulnerabilidade a corrupção de dados
- \+ Excelente desempenho
- \+ Possibilidades diversas de fluxos de trabalho

Desvantagens

- − Maior complexidade do modelo conceitual
- − Clonar um repositório é um pouco mais lento que fazer um checkout

SCV distribuído

Vantagens

- \+ Todas as vantagens de um SCV local ou com repositório centralizado
- \+ Menos vulnerabilidade a corrupção de dados
- \+ Excelente desempenho
- \+ Possibilidades diversas de fluxos de trabalho

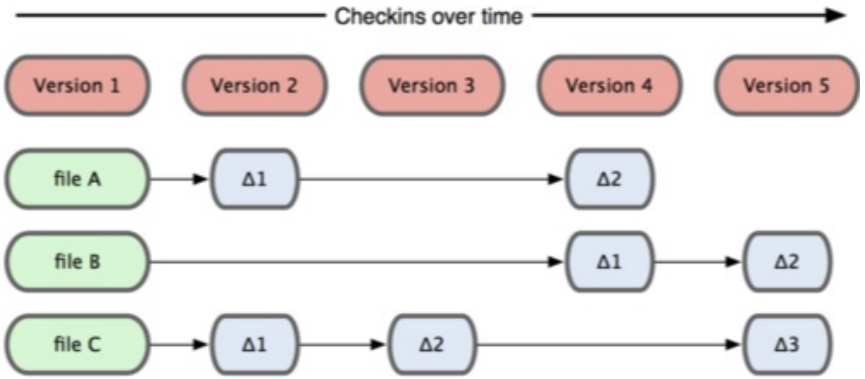
Desvantagens

- − Maior complexidade do modelo conceitual
- − Clonar um repositório é um pouco mais lento que fazer um checkout

Mais informações: http://pt.whygitisbetterthanx.com/

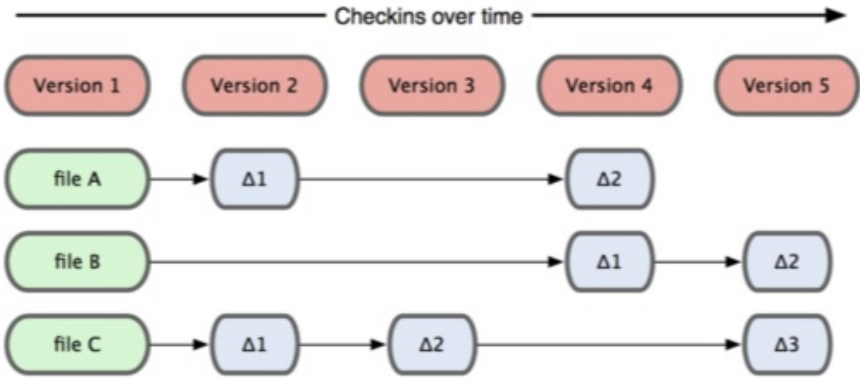
O Sistema Git

Armazenamento de versões no Git

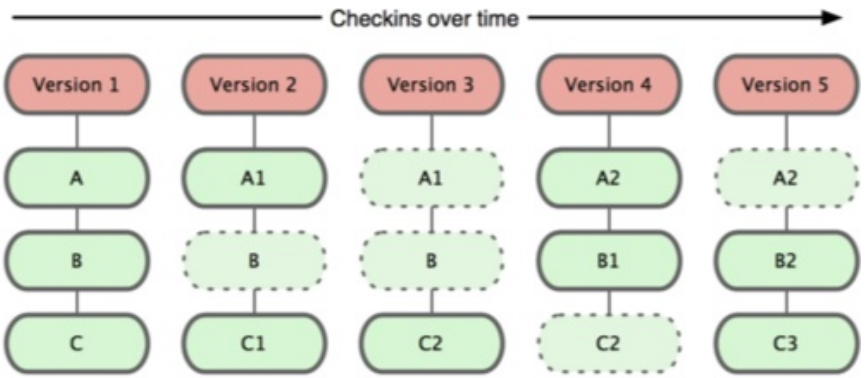


Versões armazenadas como deltas

Armazenamento de versões no Git



Versões armazenadas como deltas



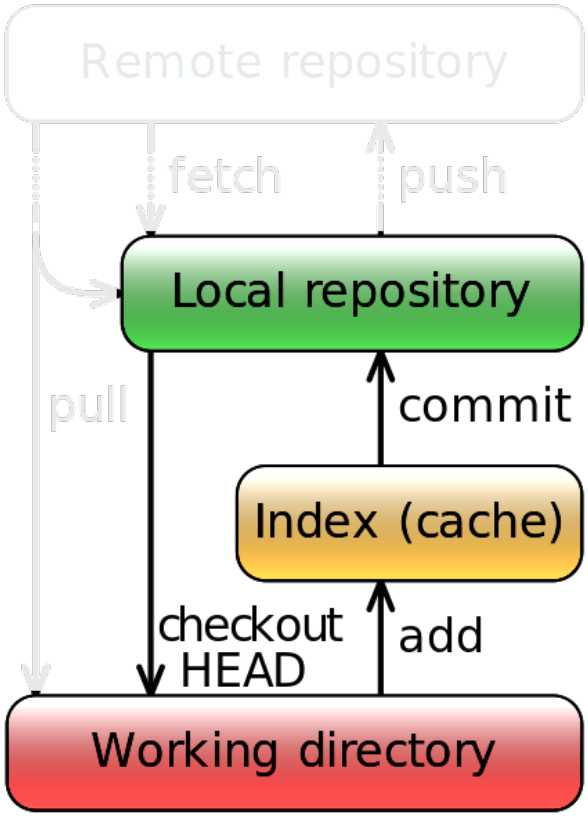
Versões armazenadas como cópias

Os três principais locais

No Git, quase tudo pode ser feito localmente

Os três principais locais

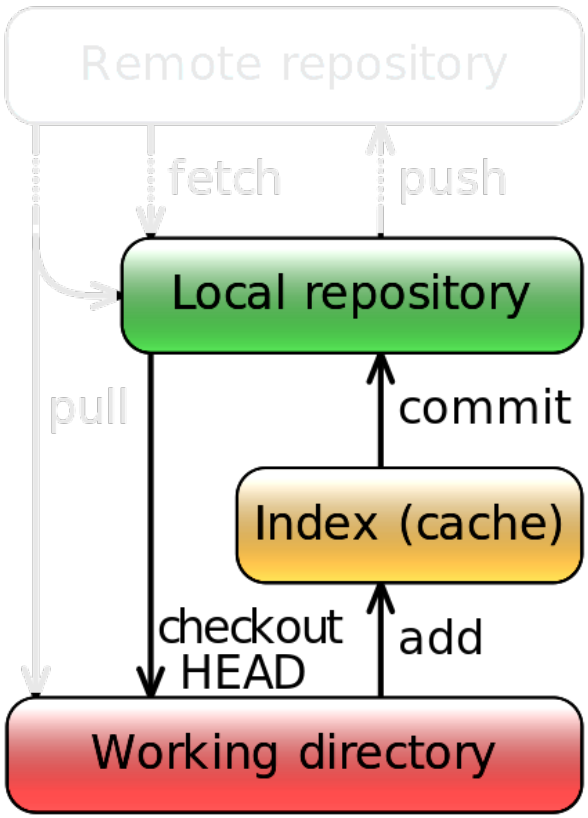
No Git, quase tudo pode ser feito localmente



http://en.wikipedia.org/wiki/File:Git_data_flow_simplified.svg

Os três principais locais

No Git, quase tudo pode ser feito **localmente**



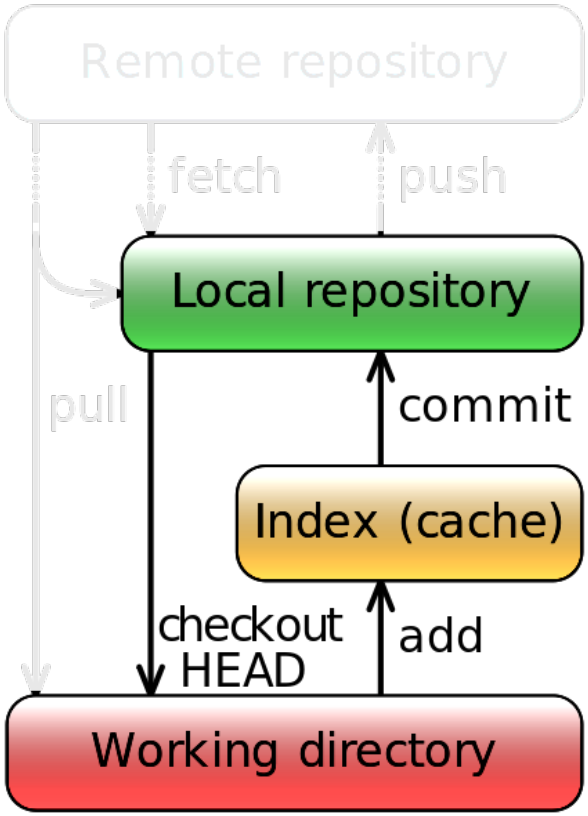
http://en.wikipedia.org/wiki/File:Git_data_flow_si

Um fluxo de trabalho

1. **add** inclui no índice arquivos **modificados** da **área de trabalho**
   - os arquivos estão **staged**
   - arquivos **não-modificados** não são incluídos

Os três principais locais

No Git, quase tudo pode ser feito localmente



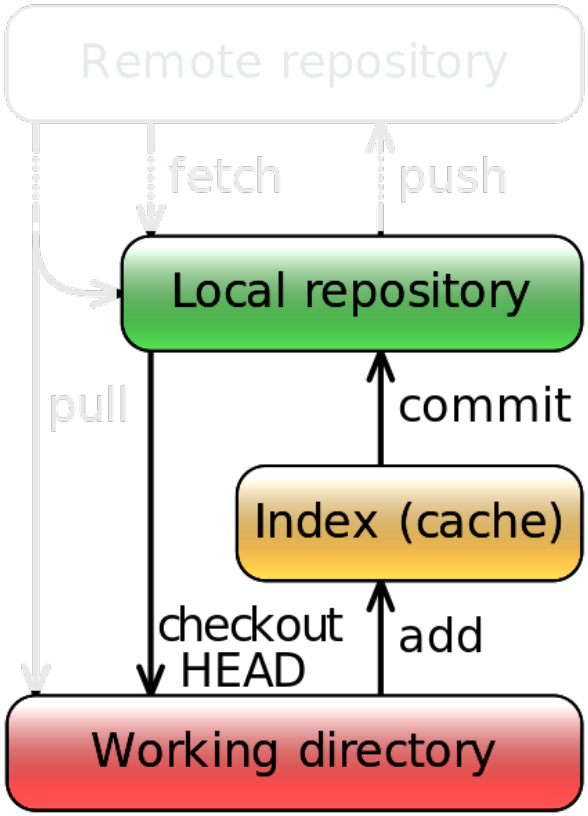
http://en.wikipedia.org/wiki/File:Git_data_flow_si

Um fluxo de trabalho

1. **add** inclui no índice arquivos **modificados** da **área de trabalho**
   - os arquivos estão **staged**
   - arquivos **não-modificados** não são incluídos
2. **commit** inclui no **repositório** itens que estão no **índice**
   - os arquivos estão **confirmados**

Os três principais locais

No Git, quase tudo pode ser feito localmente



http://en.wikipedia.org/wiki/File:Git_data_flow_si

Um fluxo de trabalho

1. **add** inclui no índice arquivos **modificados** da **área de trabalho**
   - os arquivos estão **staged**
   - arquivos **não-modificados** não são incluídos
2. **commit** inclui no **repositório** itens que estão no **índice**
   - os arquivos estão **confirmados**
3. **checkout** faz duas coisas:
   - Seleciona um outro *branch*
   - Atualiza a área de trabalho com os arquivos do *branch* selecionado

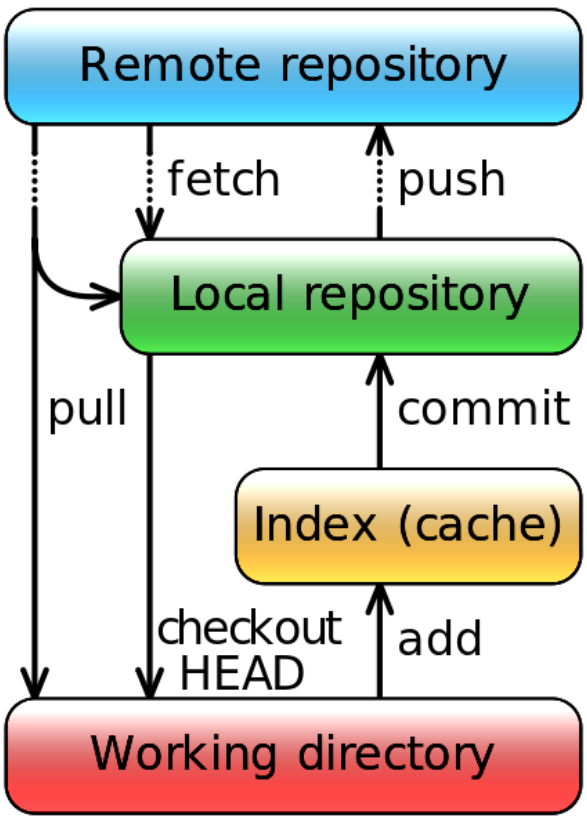
Ops! Os **quatro** principais locais

No Git, quase tudo pode ser feito localmente

Ops! Os **quatro** principais locais

No Git, quase tudo pode ser feito localmente

Às vezes, porém, tem que ser remoto

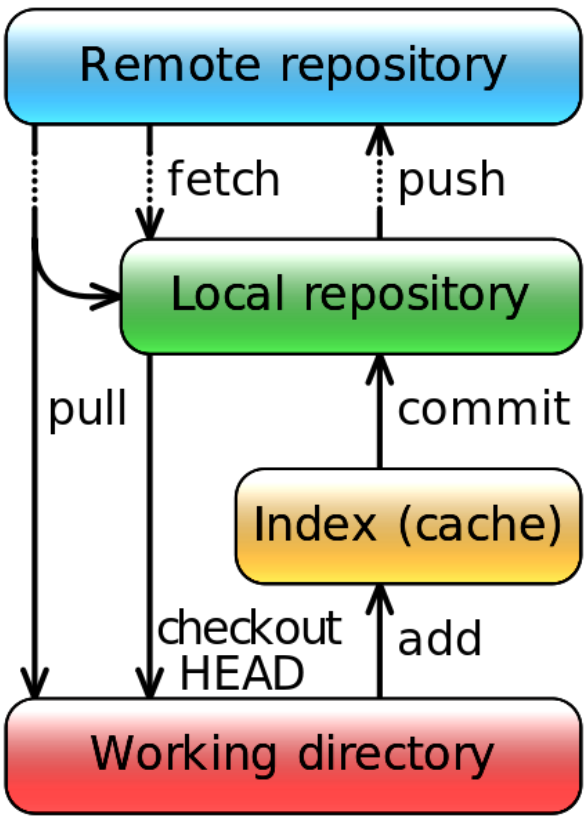


http://en.wikipedia.org/wiki/File:Git_data_flow_simplified.svg

Ops! Os **quatro** principais locais

No Git, quase tudo pode ser feito localmente

Às vezes, porém, tem que ser remoto



Um outro fluxo de trabalho

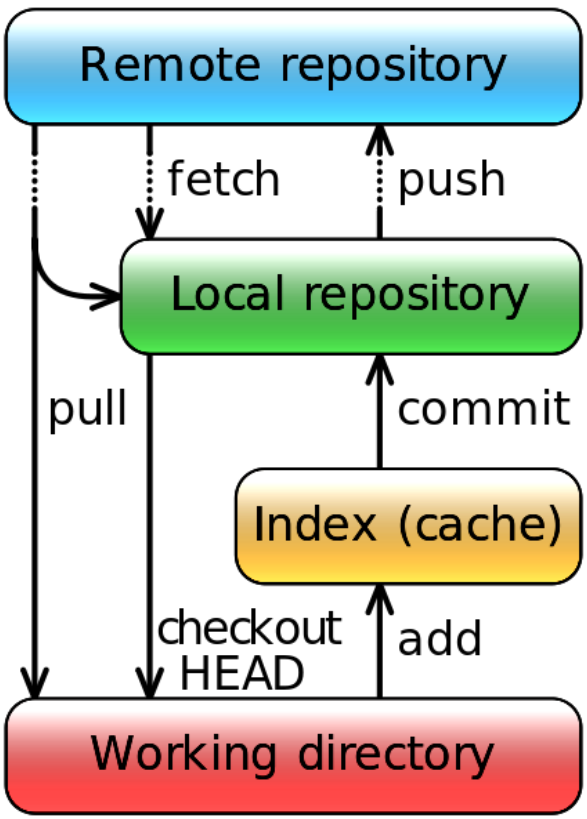
1. **push** envia modificações para o **repositório remoto**
   - para os *branches* especificados

http://en.wikipedia.org/wiki/File:Git_data_flow_simplified.svg

Ops! Os **quatro** principais locais

No Git, quase tudo pode ser feito localmente

Às vezes, porém, tem que ser remoto



Um outro fluxo de trabalho

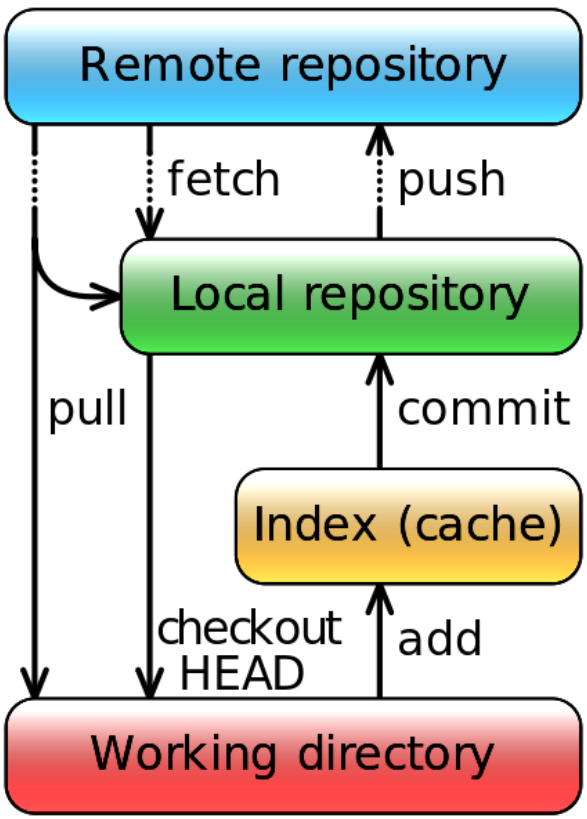
1. **push** envia modificações para o **repositório remoto**
   - para os *branches* especificados
2. **fetch** atualiza o repositório local a partir de um remoto
   - sem atualizar a area de trabalho

http://en.wikipedia.org/wiki/File:Git_data_flow_simplified.svg

Ops! Os **quatro** principais locais

No Git, quase tudo pode ser feito localmente

Às vezes, porém, tem que ser remoto



Um outro fluxo de trabalho

1. **push** envia modificações para o **repositório remoto**
   - para os *branches* especificados
2. **fetch** atualiza o repositório local a partir de um remoto
   - sem atualizar a area de trabalho
3. **pull** é similar a `fetch`, mas atualiza a área de trabalho

http://en.wikipedia.org/wiki/File:Git_data_flow_simplified.svg

Pedindo ajuda

```
$ git
usage:  ....

The most commonly used git commands are:...
    add           Add file contents to the index
...
$ git help add
```

Configurações básicas

Comando `config`

- três locais para guardar configurações:
  - repositório
  - seu diretório de usuário (opção `--global`)
  - sistema inteiro (opção `--system`)

```
$ git config user.name ‘‘Fernando Castor’’
$ git config user.email ‘‘castor@cin.ufpe.br’’
```

Criando um novo projeto

Comando `init`

```
$ pwd
/Users/fernando/scm
$ ls -1
01_conceitos_scm.ppt
02_padroes_scm.ppt
03_controle_versoes.ppt
04_gerencia_mudancas.ppt
readme.txt
$ git init

Initialized empty Git repository in /Users/fernando/scm/.git/
```

Examinando a situação atual

Comando status

```
$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   01_conceitos_scm.ppt
#   02_padroes_scm.ppt
#   03_controle_versoes.ppt
#   04_gerencia_mudancas.ppt
#   readme.txt

nothing added to commit but untracked files present (use "git add"to

track)
```

Examinando a situação atual

Comando status

```
$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   01_conceitos_scm.ppt
#   02_padroes_scm.ppt
#   03_controle_versoes.ppt
#   04_gerencia_mudancas.ppt
#   readme.txt

nothing added to commit but untracked files present (use "git add"to

track)
```

Notem o ‘use "git add"...’

Rastreando arquivos

Comando add

```
$ git add readme.txt
$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#   new file:  readme.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   01_conceitos_scm.ppt
...nothing added to commit but untracked files present (use "git add"to
track)
```

Fazendo *commit* para o repositório

Comando commit

```
$ git commit -m "Arquivo de instruções."
  1 files changed, 1 insertions(+), 0 deletions(-)
  create mode 100644 readme.txt

(modificamos um pouco mais o arquivo readme.txt)

$ git commit -m "O novo readme.txt."
[master 24167e8] O novo readme.txt.
  1 files changed, 4 insertions(+), 1 deletions(-)
```

Histórico de *commits*

Comando `log`

```
$ git log --stat
commit 24167e85de618bf9240cf14b51e13195b2218a14
Author:  Fernando Castor <castor@cin.ufpe.br>
Date:  Tue Apr 24 17:31:35 2012 -0300

      O novo readme.txt.

  readme.txt |      5 ++++-
  1 files changed, 4 insertions(+), 1 deletions(-)

commit 67fd964ad19b9a59d531f2bd6e12419c2ff189e1
Author:  Fernando Castor <castor@cin.ufpe.br>
Date:  Tue Apr 24 17:21:08 2012 -0300

      Arquivo de instruções.

  readme.txt |      1 +
  1 files changed, 1 insertions(+), 0 deletions(-)
```

Verificando diferenças entre arquivos

Comando `diff`

(modificamos um pouco mais o arquivo `readme.txt`)

```
$ git diff
diff --git a/readme.txt b/readme.txt
index 44c8c80..012e742 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1,4 +1,7 @@
This is now so much more than a readme file.  It is something
-altogether different.  I know, this is still a readme file.  However,
...
+altogether different.  I know, this is still a readme file.  It does

...
```

Ainda verificando diferenças entre arquivos

(continuando do slide anterior)

```
$ git add readme.txt
$ git diff
$
$ git diff --cached
index 44c8c80..012e742 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1,4 +1,7 @@
...
```

Criando *branches*

Comando `branch`

```
$ git branch b1
$ git branch
  b1
* master
```

Criando *branches*

Comando branch

```
$ git branch b1
$ git branch
  b1
* master
```

```
(cria-se um arquivo apenasNoBranchMaster.txt)

$ ls -1
...
04_gerencia_mudancas.ppt
apenasNoBranchMaster.txt
readme.txt
```

Criando *branches*

Comando branch

```
$ git branch b1
$ git branch
  b1
* master
```

```
(cria-se um arquivo apenasNoBranchMaster.txt)

$ ls -1
...
04_gerencia_mudancas.ppt
apenasNoBranchMaster.txt
readme.txt
$ git checkout b1
$ ls -1
...
04_gerencia_mudancas.ppt
readme.txt
```

Combinando alterações em *branches*

Comando merge

```
(depois de fazer alterações em readme.txt nos dois branches)

$ git checkout master
$ git merge b1
Auto-merging readme.txt
CONFLICT (content):  Merge conflict in readme.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Como conflitos são representados em arquivos texto

```
<<<<<<< HEAD
altogether different.  I know, this is still a readme file.  It does
not have many of the features of a readme file, though.  In fact, I
find it lacking in almost any aspect I can think of.  However, it has
become my example file to show the student the amazing capabilities of
Git.  So, I'll use it anyway.

=======
altogether different.  I know, this is still a readme file.
Nonetheless,
it has become my example file to show the student the amazing
Just inserting a line her to make things more complex.
capabilities of Git.
I think I understand what can make conflicts occur.

>>>>>>> b1
```

Como conflitos são representados em arquivos texto

```
<<<<<<< HEAD
altogether different.  I know, this is still a readme file.  It does
not have many of the features of a readme file, though.  In fact, I
find it lacking in almost any aspect I can think of.  However, it has
become my example file to show the student the amazing capabilities of
Git.  So, I'll use it anyway.

=======
altogether different.  I know, this is still a readme file.
Nonetheless,
it has become my example file to show the student the amazing
Just inserting a line her to make things more complex.
capabilities of Git.
I think I understand what can make conflicts occur.

>>>>>>> b1
```

Depois de resolver os conflitos, é só usar `add` e `commit`.

Como conflitos são representados em arquivos texto

```
<<<<<<< HEAD
altogether different.  I know, this is still a readme file.  It does
not have many of the features of a readme file, though.  In fact, I
find it lacking in almost any aspect I can think of.  However, it has
become my example file to show the student the amazing capabilities of
Git.  So, I'll use it anyway.

=======
altogether different.  I know, this is still a readme file.
Nonetheless,
it has become my example file to show the student the amazing
Just inserting a line her to make things more complex.
capabilities of Git.
I think I understand what can make conflicts occur.

>>>>>>> b1
```

Depois de resolver os conflitos, é só usar `add` e `commit`.

Para ver melhor o histórico, use a ferramenta `gitk`

Clonando um repositório remoto

Comando `clone`

```
git clone protocolo://endereco/caminho/repositorio.git
```

Clonando um repositório remoto

Comando `clone`

```
git clone protocolo://endereco/caminho/repositorio.git
```

Por exemplo:

```
git clone git://github.com/curso-scm/scm.git  ou

git clone https://github.com/curso-scm/scm.git
```

Clonando um repositório remoto

Comando `clone`

```
git clone protocolo://endereco/caminho/repositorio.git
```

Por exemplo:

```
git clone git://github.com/curso-scm/scm.git  ou

git clone https://github.com/curso-scm/scm.git
```

- Não funciona como em SCVs com repositório centralizado
- A princípio, não há um repositório mais importante

Enviando arquivos atualizados para um repositório remoto

Comando `push`

```
git push https://github.com/curso-scm/scm.git master
```

Enviando arquivos atualizados para um repositório remoto

Comando `push`

```
git push https://github.com/curso-scm/scm.git master
```

O segundo parâmetro pode ser

- Omitido.
- Um arquivo (ou vários)
- Um branch
- A chave SHA1 de um objeto
- Entre outras coisas...

Trazendo objetos para repositório local

Comando `pull`

```
git pull https://github.com/curso-scm/scm.git master
```

Trazendo objetos para repositório local

Comando `pull`

```
git pull https://github.com/curso-scm/scm.git master
```

O segundo parâmetro pode ser

- Omitido.
- Um arquivo (ou vários)
- Um branch
- A chave SHA1 de um objeto
- Entre outras coisas...

Trazendo objetos para repositório local

Comando `pull`

```
git pull https://github.com/curso-scm/scm.git master
```

O segundo parâmetro pode ser

- Omitido.
- Um arquivo (ou vários)
- Um branch
- A chave SHA1 de um objeto
- Entre outras coisas...

O comando `fetch` pode ser usado também

- Não realiza `merge`.

Hospedagem centrada em Git: Github e Gitorious

Hospedagem gratuita de projetos

O que é?

- Serviço oferecido por alguns sítios
  - Exemplos: SourceForge, GoogleCode, Codeplex.
- Mantém os arquivos acessíveis remotamente em um repositório
- Em geral, dão suporte a várias ferramentas

Hospedagem gratuita de projetos

O que é?

- Serviço oferecido por alguns sítios
  - Exemplos: SourceForge, GoogleCode, Codeplex.
- Mantém os arquivos acessíveis remotamente em um repositório
- Em geral, dão suporte a várias ferramentas

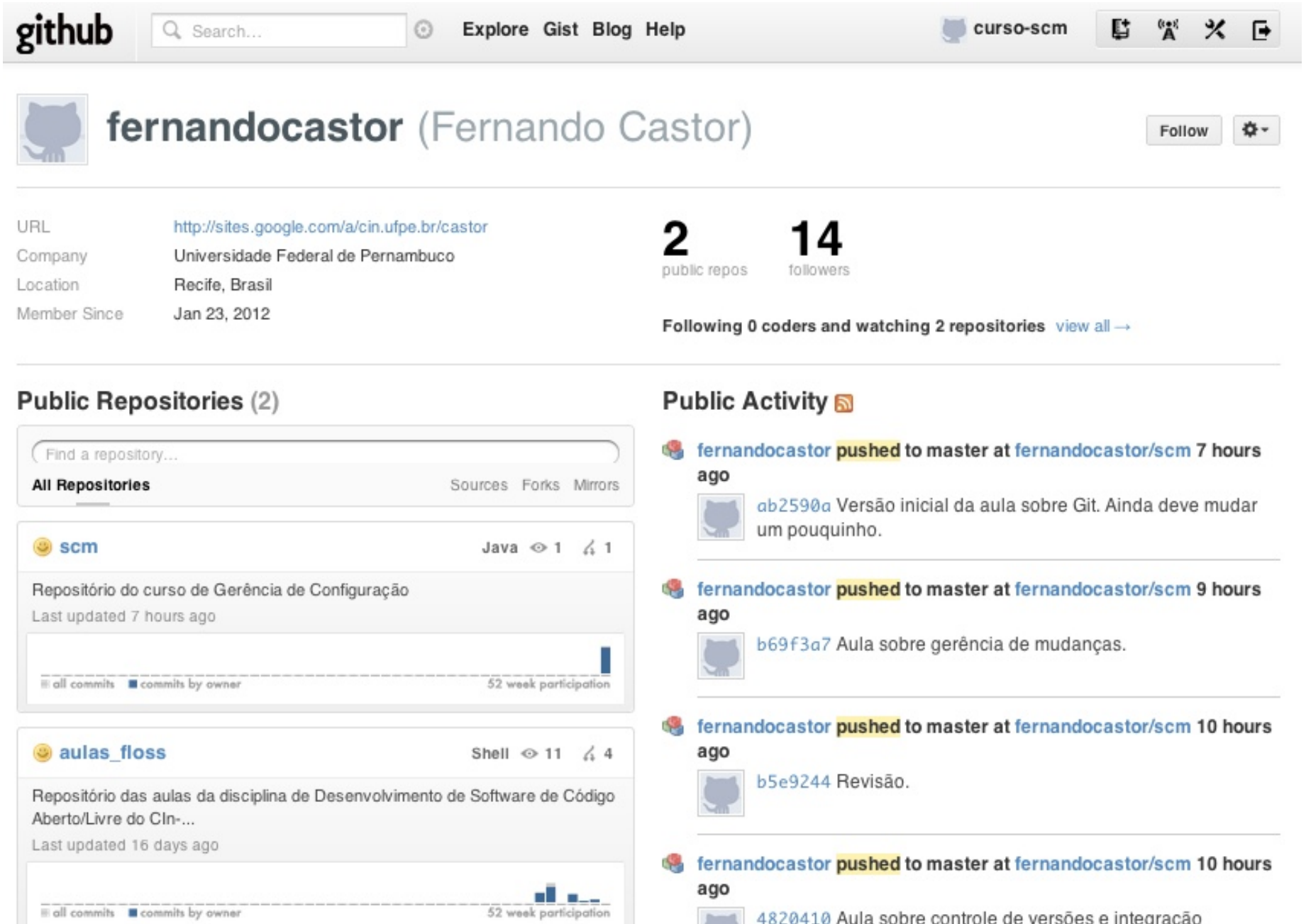
Hospedagem centrada no Git

- Dois exemplos mais conhecidos:
  - GitHub e Gitorious.
- Lembram redes sociais

GitHub

Funcionalidades

- Criação e manutenção de vários repositórios públicos
- Registro de diversas estatísticas de uso
- Possibilidade de "seguir" outros desenvolvedores


github
Search...
Explore Gist Blog Help
curso-scm
fernandocastor (Fernando Castor)
Follow
URL http://sites.google.com/a/cin.ufpe.br/castor
Company Universidade Federal de Pernambuco
Location Recife, Brasil
Member Since Jan 23, 2012
2 public repos
14 followers
Following 0 coders and watching 2 repositories view all →
Public Repositories (2)
Find a repository...
All Repositories Sources Forks Mirrors
scm Java 1 1
Repositório do curso de Gerência de Configuração
Last updated 7 hours ago
all commits commits by owner 52 week participation
aulas_floss Shell 11 4
Repositório das aulas da disciplina de Desenvolvimento de Software de Código Aberto/Livre do CIn-...
Last updated 16 days ago
all commits commits by owner 52 week participation
Public Activity
fernandocastor pushed to master at fernandocastor/scm 7 hours ago
ab2590a Versão inicial da aula sobre Git. Ainda deve mudar um pouquinho.
fernandocastor pushed to master at fernandocastor/scm 9 hours ago
b69f3a7 Aula sobre gerência de mudanças.
fernandocastor pushed to master at fernandocastor/scm 10 hours ago
b5e9244 Revisão.
fernandocastor pushed to master at fernandocastor/scm 10 hours ago
4820410 Aula sobre controle de versões e integração

Obrigado!


Centro
de Informática
U·F·P·E

ines

SPG
SOFTWARE·PRODUCTIVITY·GROUP