



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
BAHIA



Instituto Federal da Bahia
Análise e Desenvolvimento de Sistemas
INF022 – Tópicos Avançados

Software Configuration Management

Prof. Dr. Renato L. Novais
renato@ifba.edu.br

Agenda



- CVS/SVN
- GIT

Contextualização



- Equipe composta por mais de uma pessoa
- Sincronização de código conflitante
- Várias versões
- *Backup*
- Código pertence a todos
- *Bugs* inseridos depois de uma pequena modificação

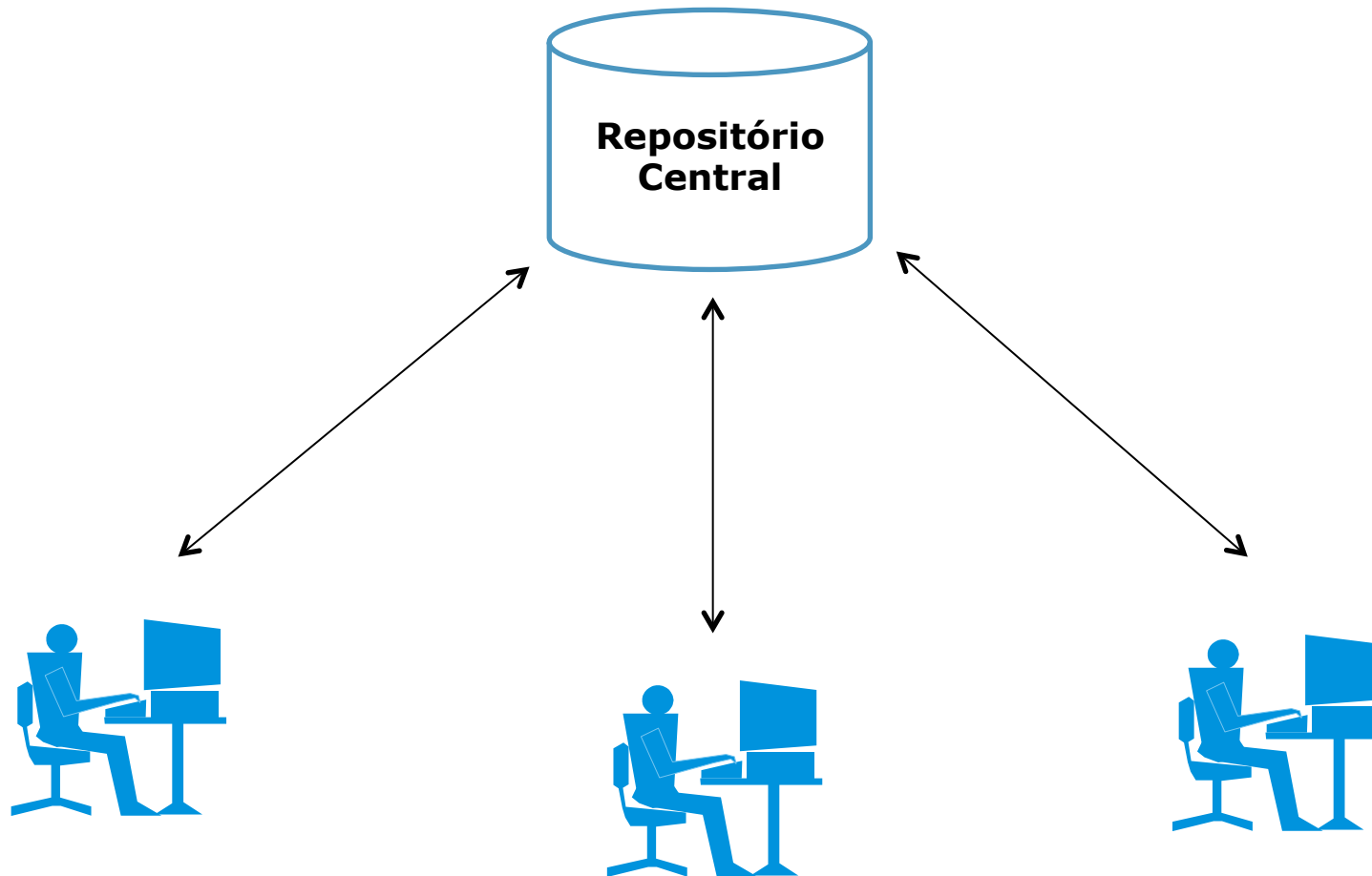
Concurrent Versions System



- O CVS é sistema de controle de versão;
- Gerência de Configuração de código;
- Permite que pessoas dispersas trabalhem no mesmo código.
- *"CVS is not a substitute for management... for developer communication. "*

- CVS utiliza uma **arquitetura cliente-servidor**:
 - um servidor armazena a(s) versão(ões) atuais do projeto e seu histórico;
 - clientes se conectam a esse servidor para obter uma cópia completa do projeto, trabalhar nessa cópia e então devolver suas modificações.
- Tipicamente, cliente e servidor devem estar conectados por uma rede local de computadores;
- Vários clientes podem editar cópias do mesmo projeto de maneira concorrente. Quando eles confirmam suas alterações, o servidor tenta fazer uma fusão delas;

SVN – Visão Geral



Terminologia



- *Release*: é a versão de um produto inteiro.
- *Revision*: é a numeração atribuída pelo CVS a cada modificação de um arquivo.
- O *Checkout*: é usado para denominar o primeiro *download* de um módulo inteiro a partir do repositório CVS.
- *Commit*: envio das modificações feitas pelo usuário ao repositório CVS.

Terminologia



- **Export:** é o *download* de um módulo inteiro a partir de um repositório CVS, sem os arquivos administrativos CVS. Módulos exportados não ficam sob controle do CVS.
- **Import:** é usado para designar a criação de um módulo inteiro dentro de um repositório CVS através do *upload* de uma estrutura de diretórios.
- **Branch:** é uma ramificação no desenvolvimento, usada para descrever o processo de divisão dos arquivos de um projeto em linhas de desenvolvimento independentes. Podendo servir para teste de uma nova funcionalidade ou para projetos destinados a um cliente específico.

Terminologia

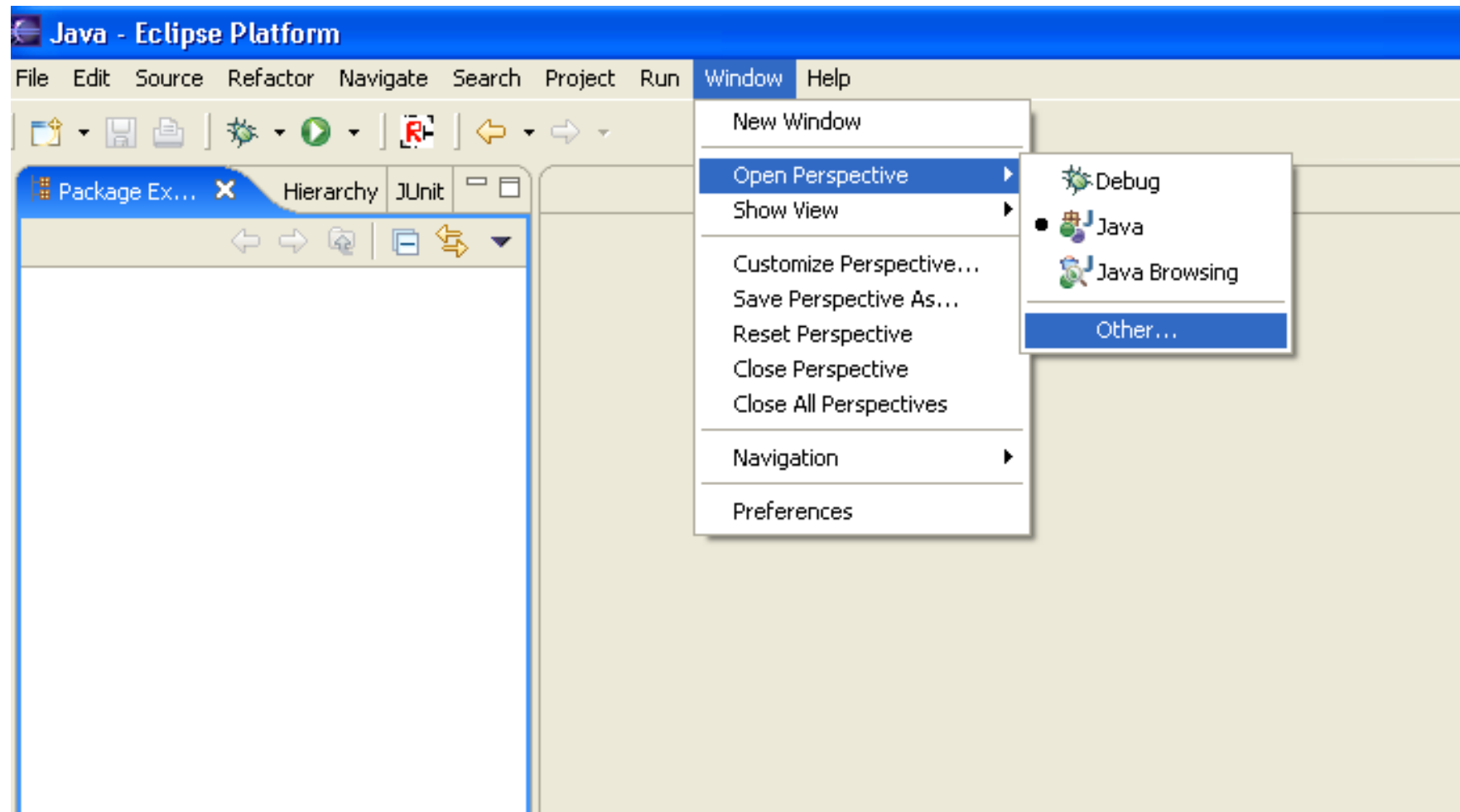


- *Update*: atualização da cópia local do trabalho através do download das modificações feitas por outros usuários no repositório.
- *Merge*: é a fusão de modificações feitas por diferentes usuários na cópia local de um mesmo arquivo. Sempre que alguém altera o código, é necessário realizar um update antes do commit, de modo que seja feito o merge — ou a fusão — das mudanças.

CVS integrado ao eclipse

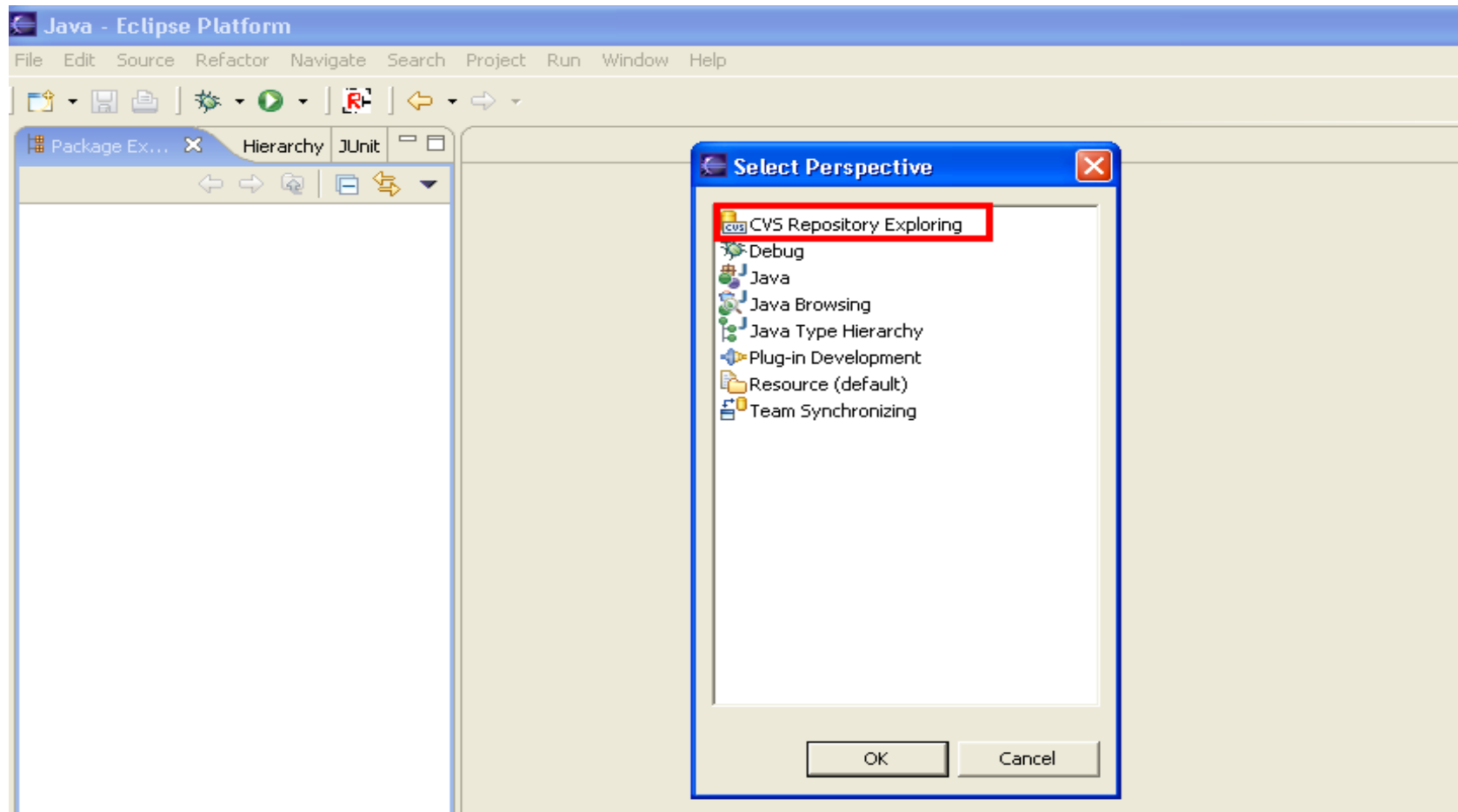


- Selecionar perspectiva do CVS



CVS integrado ao eclipse

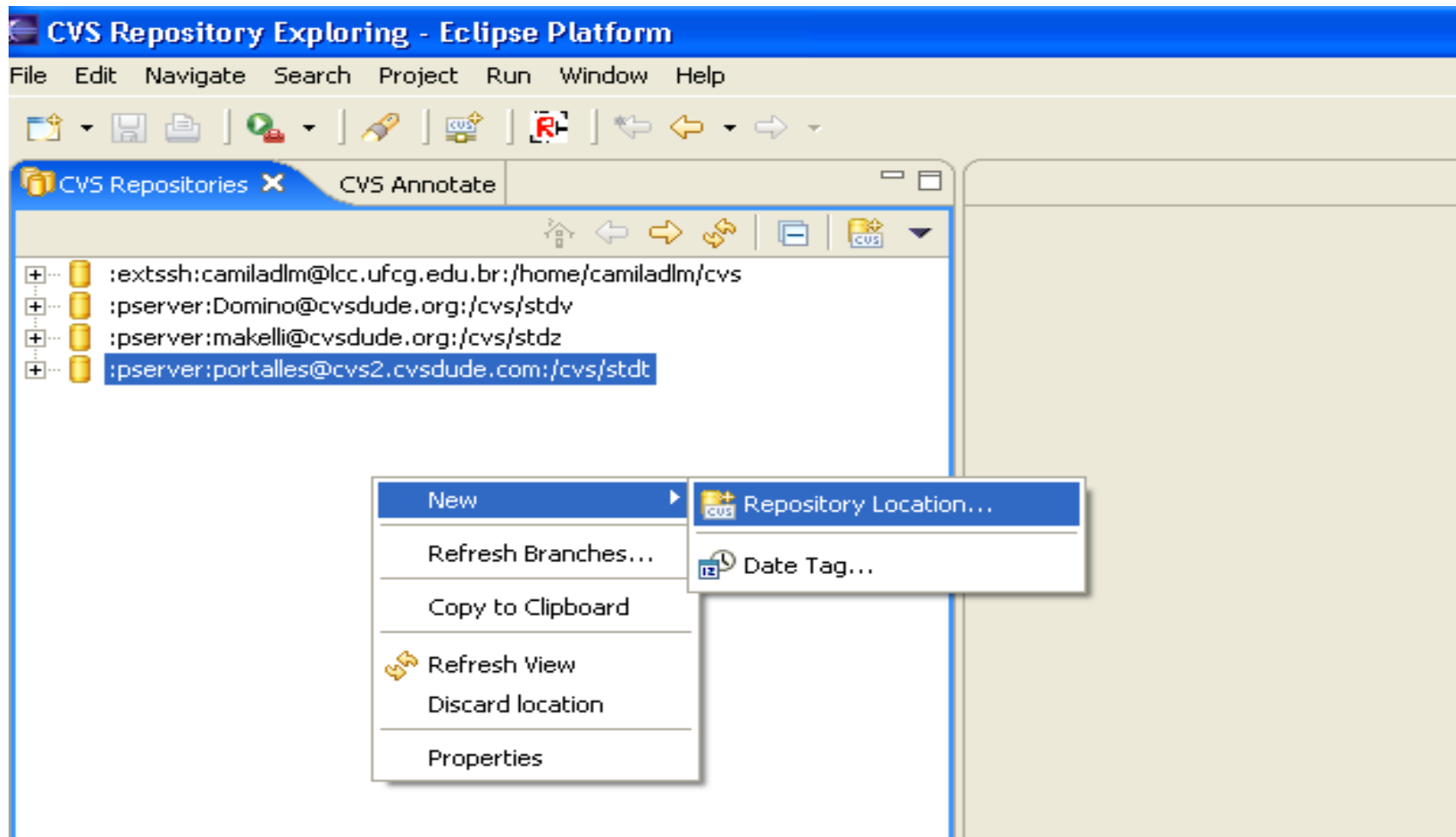
- Selecionar perspectiva do CVS



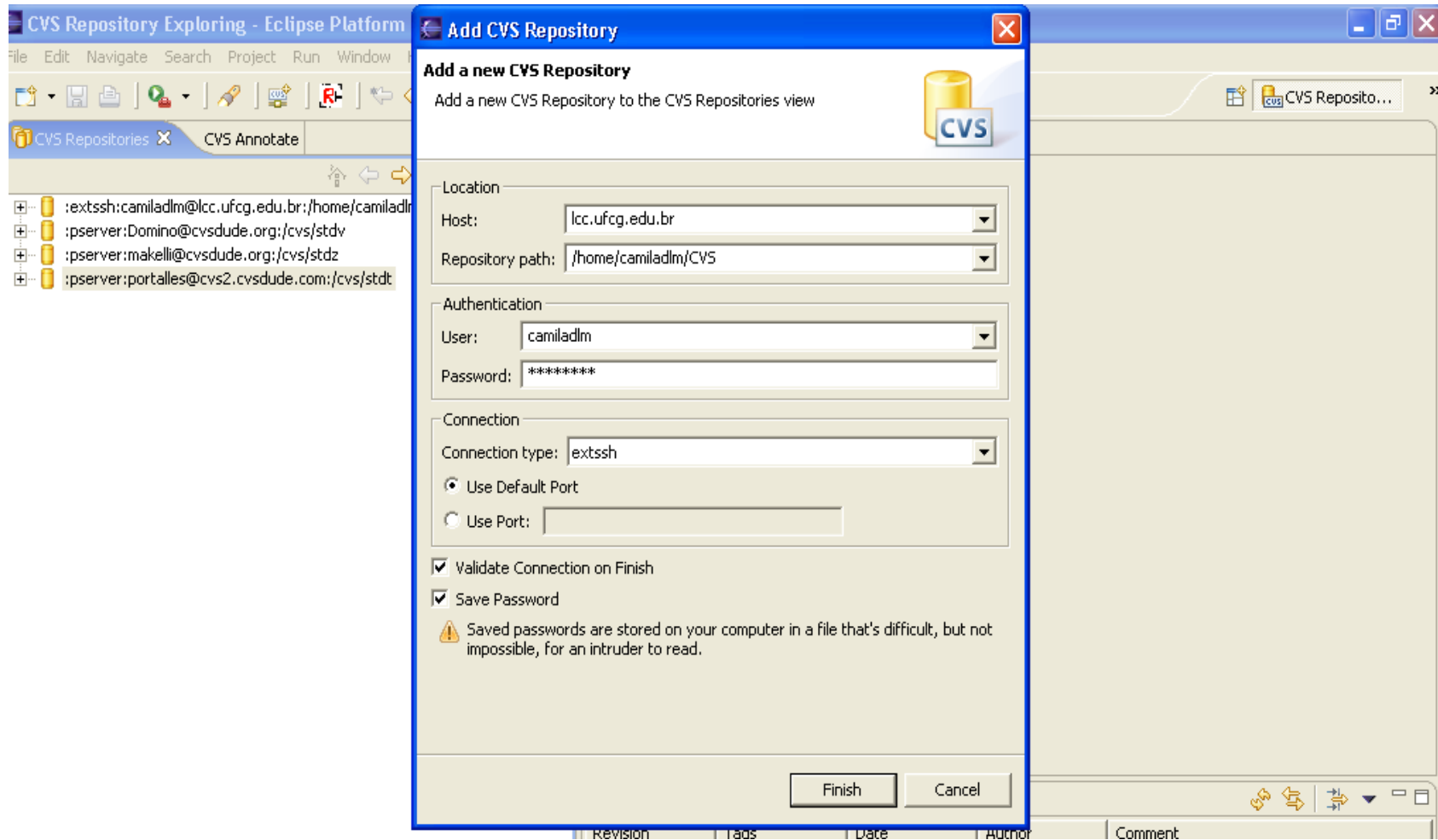
CVS integrado ao eclipse



- Adicionar um novo Repositório



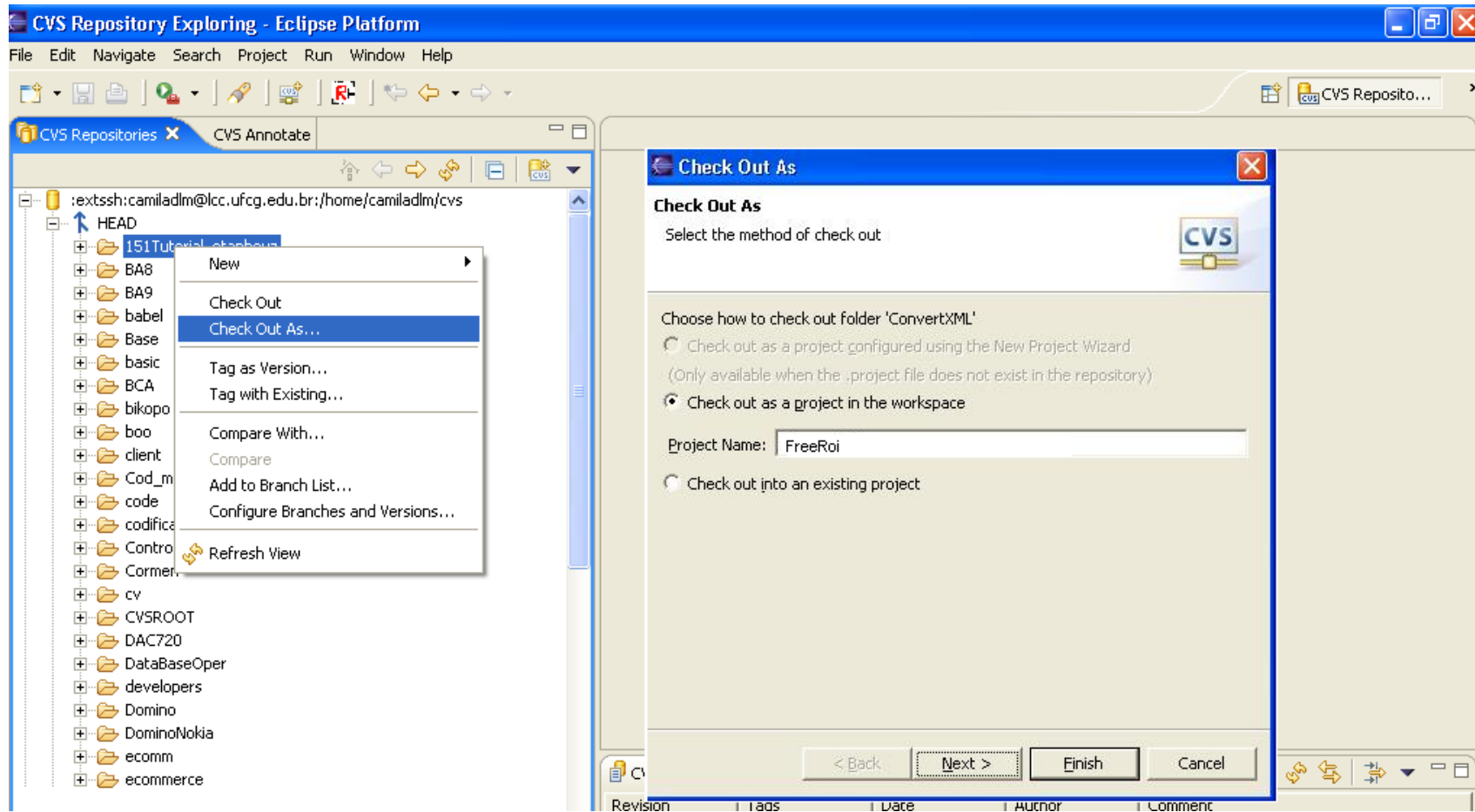
- Adicionar um novo Repositório



CVS integrado ao eclipse



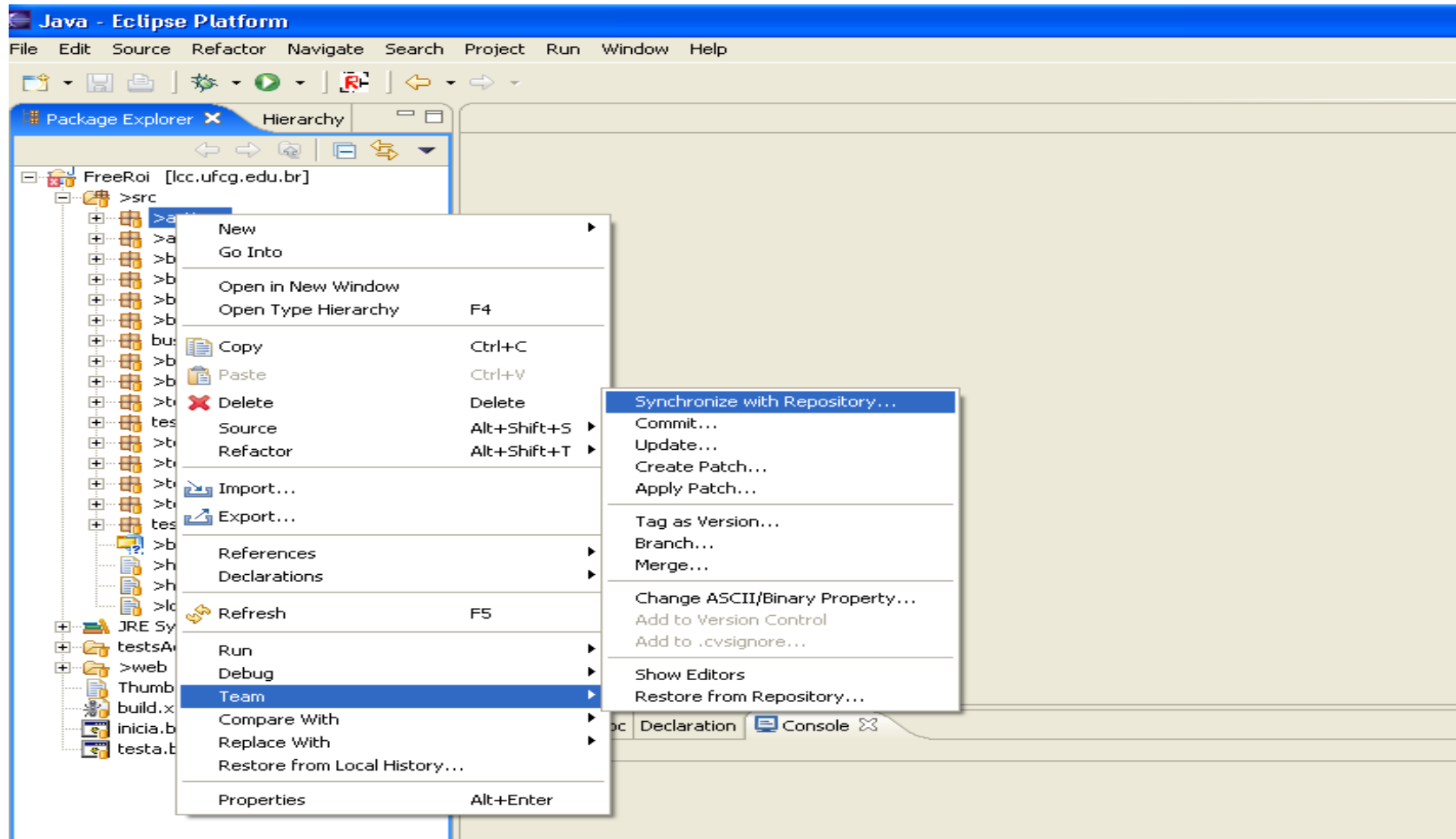
- Fazer um checkout



CVS integrado ao eclipse



- Sincronizando...

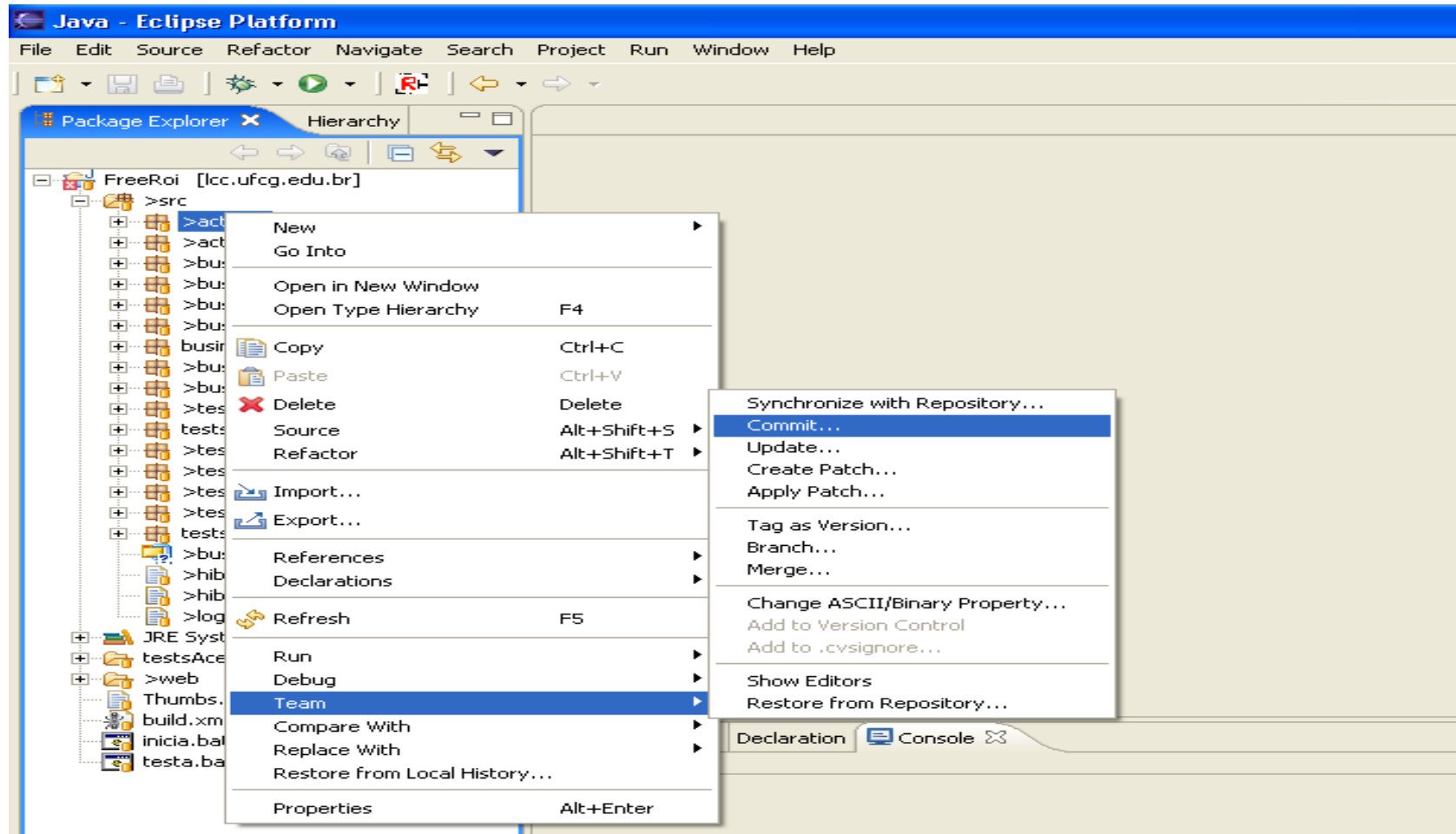


CVS integrado ao eclipse



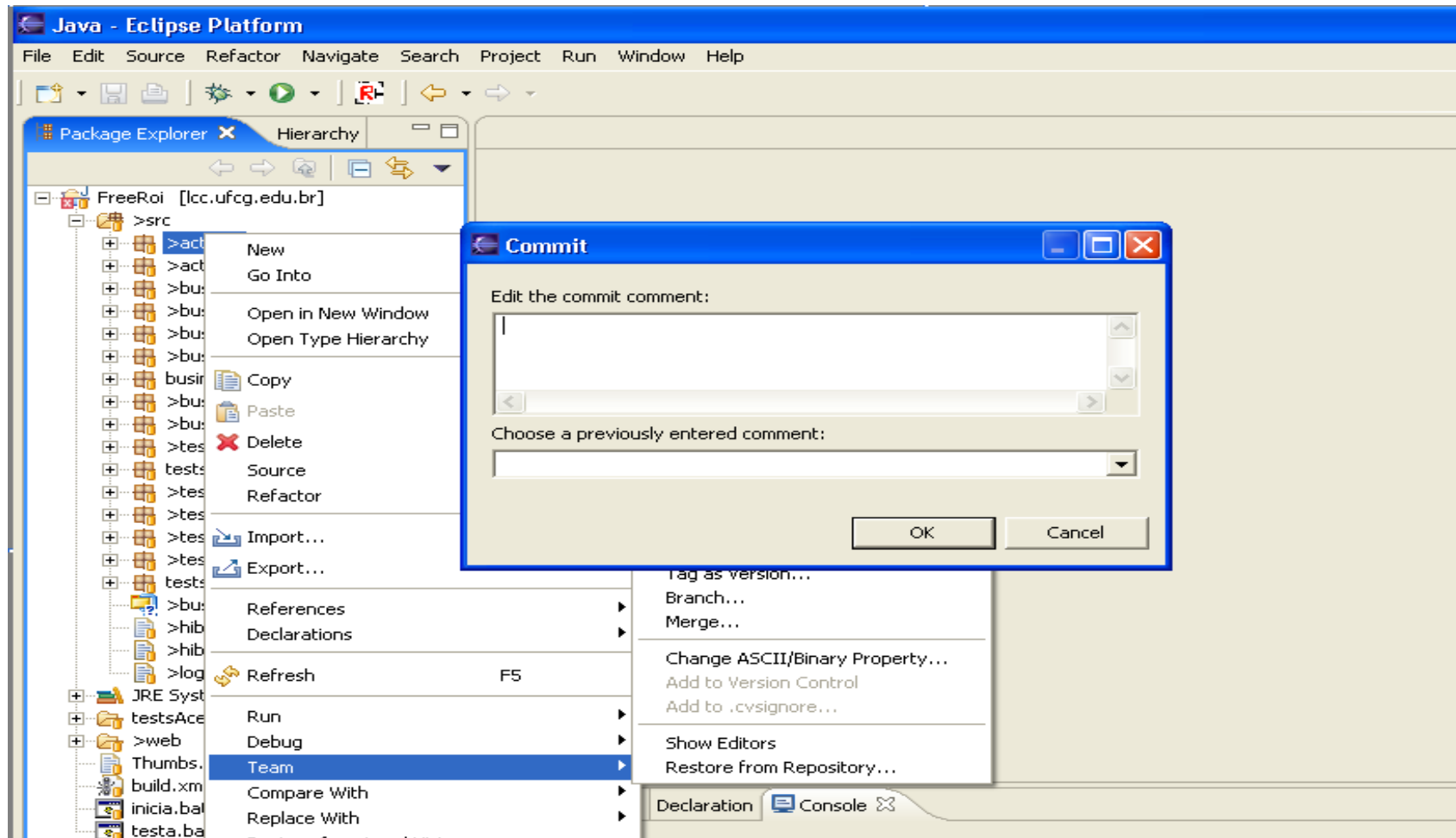
- *Commit*

Análise e Desenvolvimento de Sistemas



CVS integrado ao eclipse

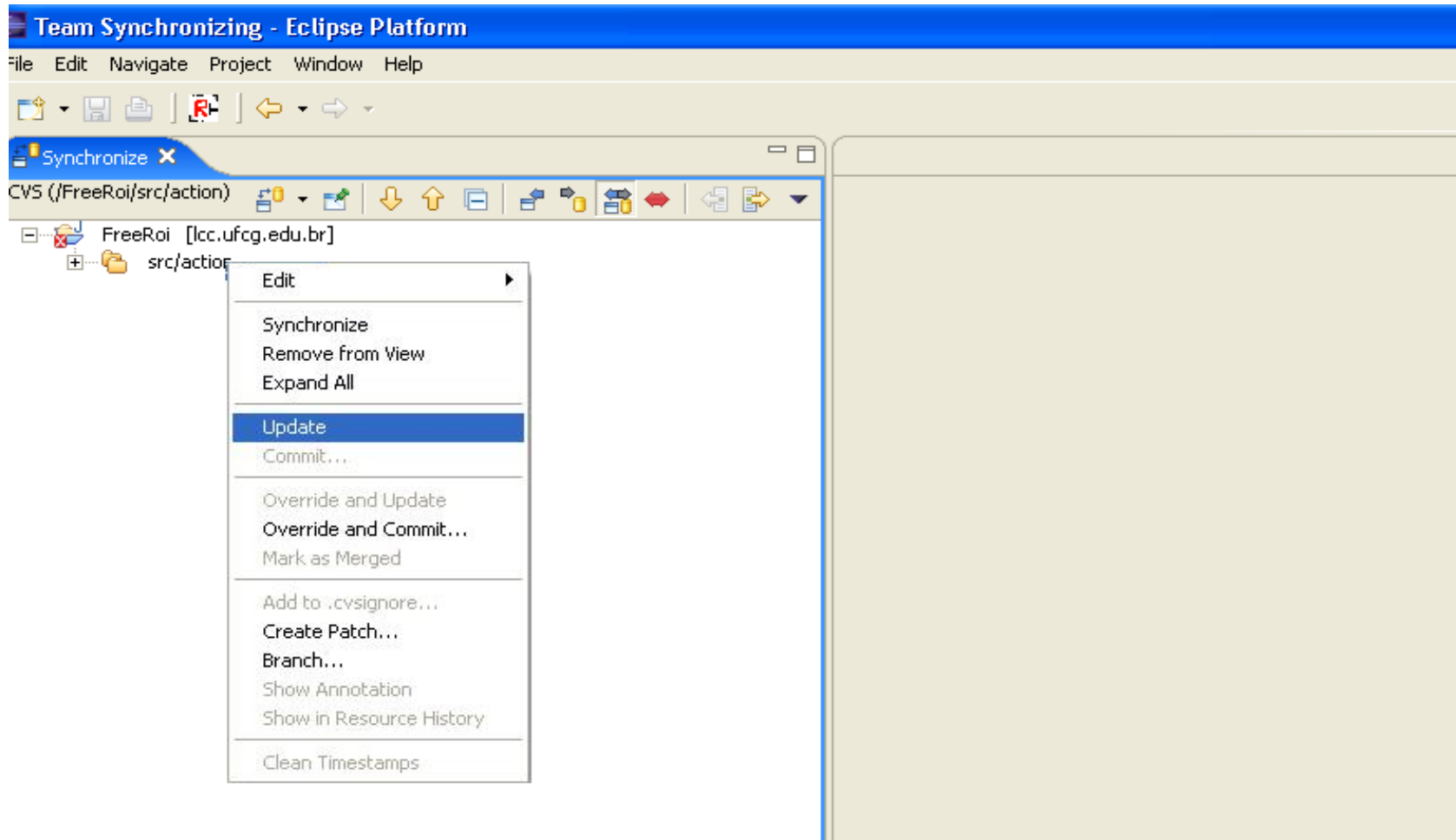
- *Commit*



CVS integrado ao eclipse



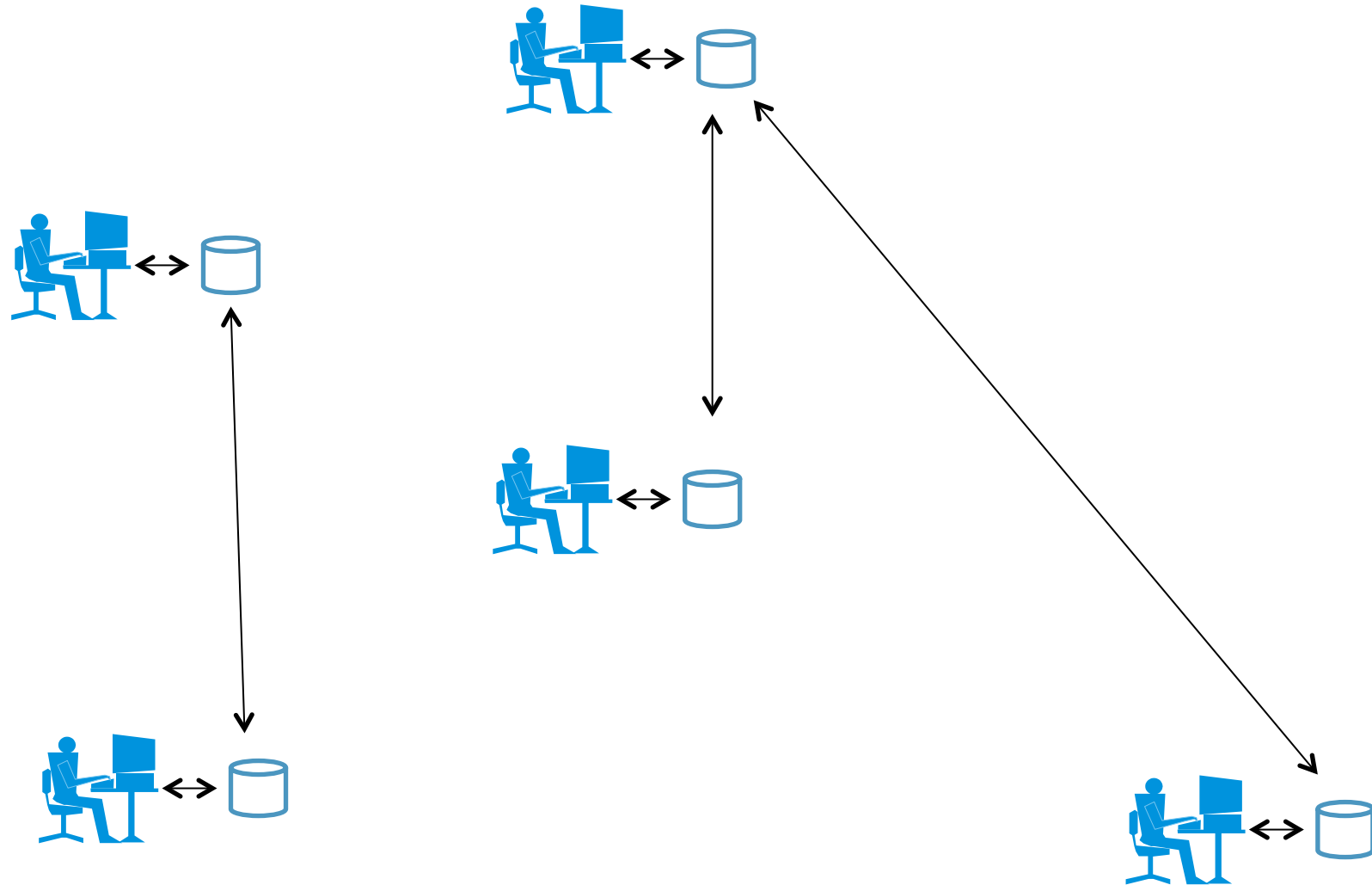
- Update



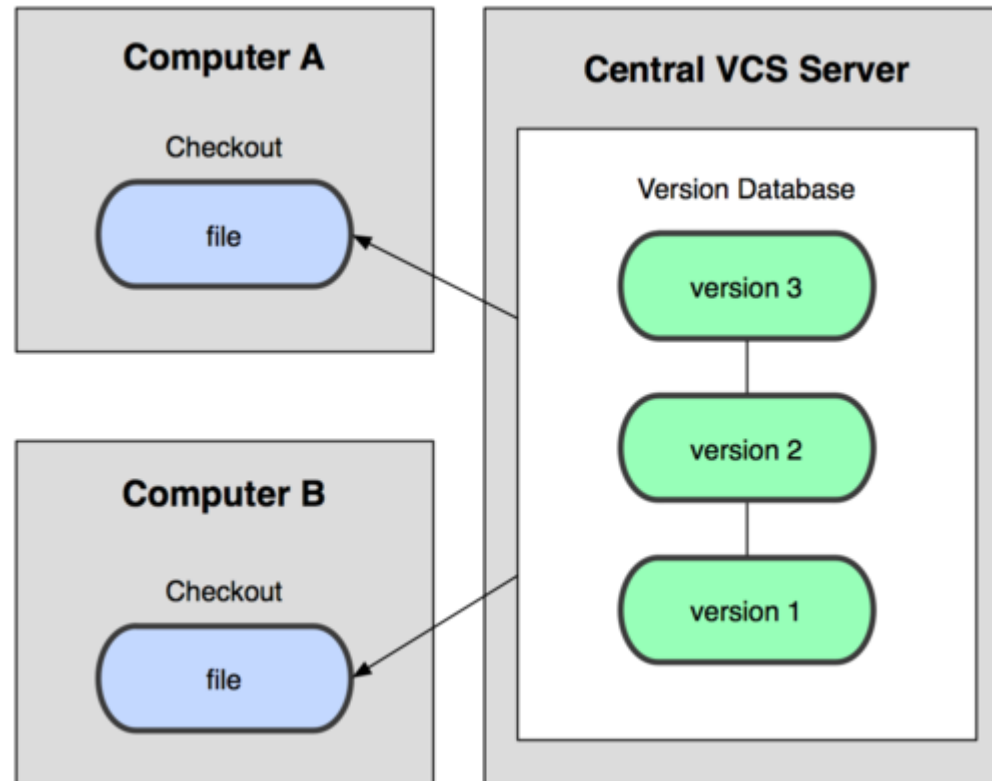
- Subversion (também conhecido por svn) é um sistema de controle de versão que tem ganhado bastante adeptos nos últimos tempos. Pretende-se ser mais robusto que o CVS.
<http://subversion.tigris.org>
- Atualmente utiliza protocolo HTTP, rodando sobre o apache
- Cliente Tortoise SVN (stand alone)
<http://tortoisesvn.tigris.org>
- Plugins para o Eclipse, Netbeans

- O linux usava o BitKeeper (proprietário, tinha uma licença isento-de-pagamento)
- Em 2005 a parceria foi quebrada
- Os desenvolvedores do linux (especialmente Linux Torvalds) resolveram fazer uma ferramenta baseada no conhecimentos adquiridos do BitKeeper
- Objetivos
 - Velocidade
 - Design simples
 - Suporte robusto a desenvolvimento não linear (milhares de branches paralelos)
 - Totalmente distribuído
 - Capaz de lidar eficientemente com grandes projetos como o kernel do Linux (velocidade e volume de dados)

GIT – Visão Geral

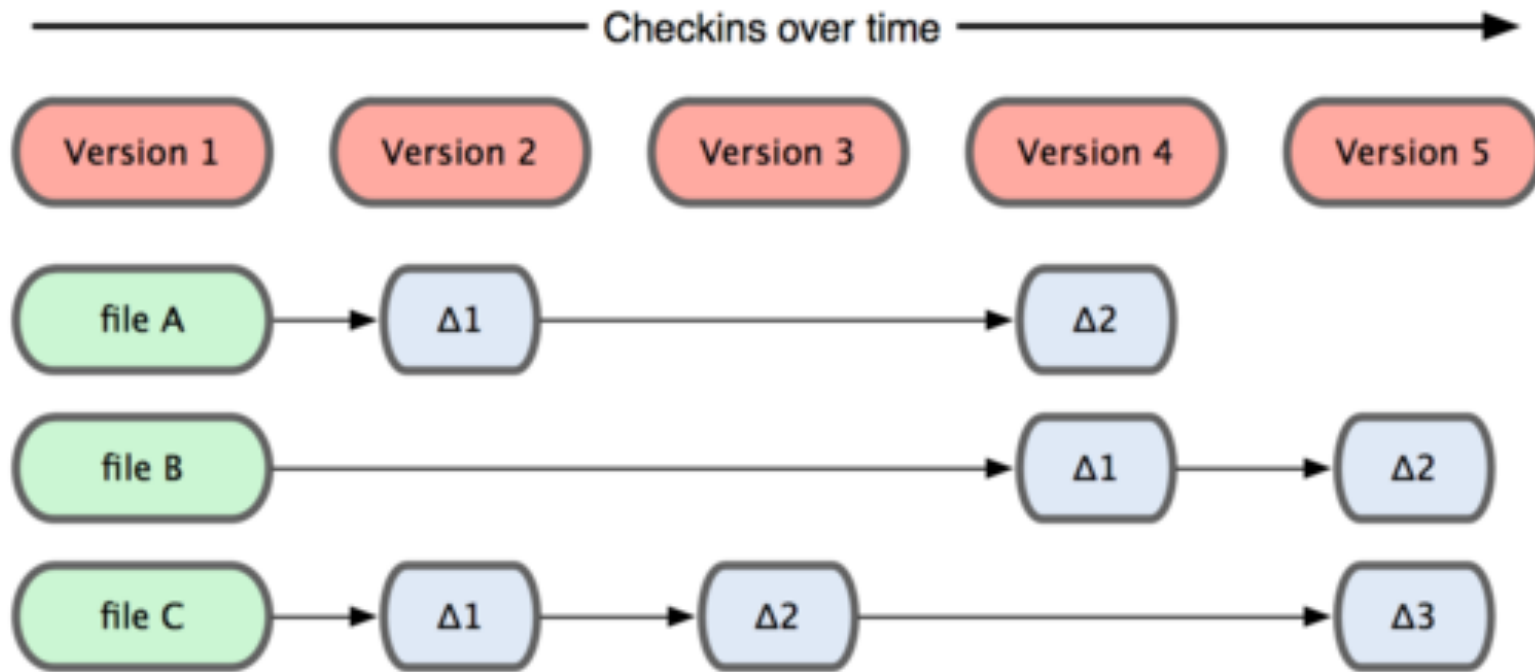


Centralized Version Control

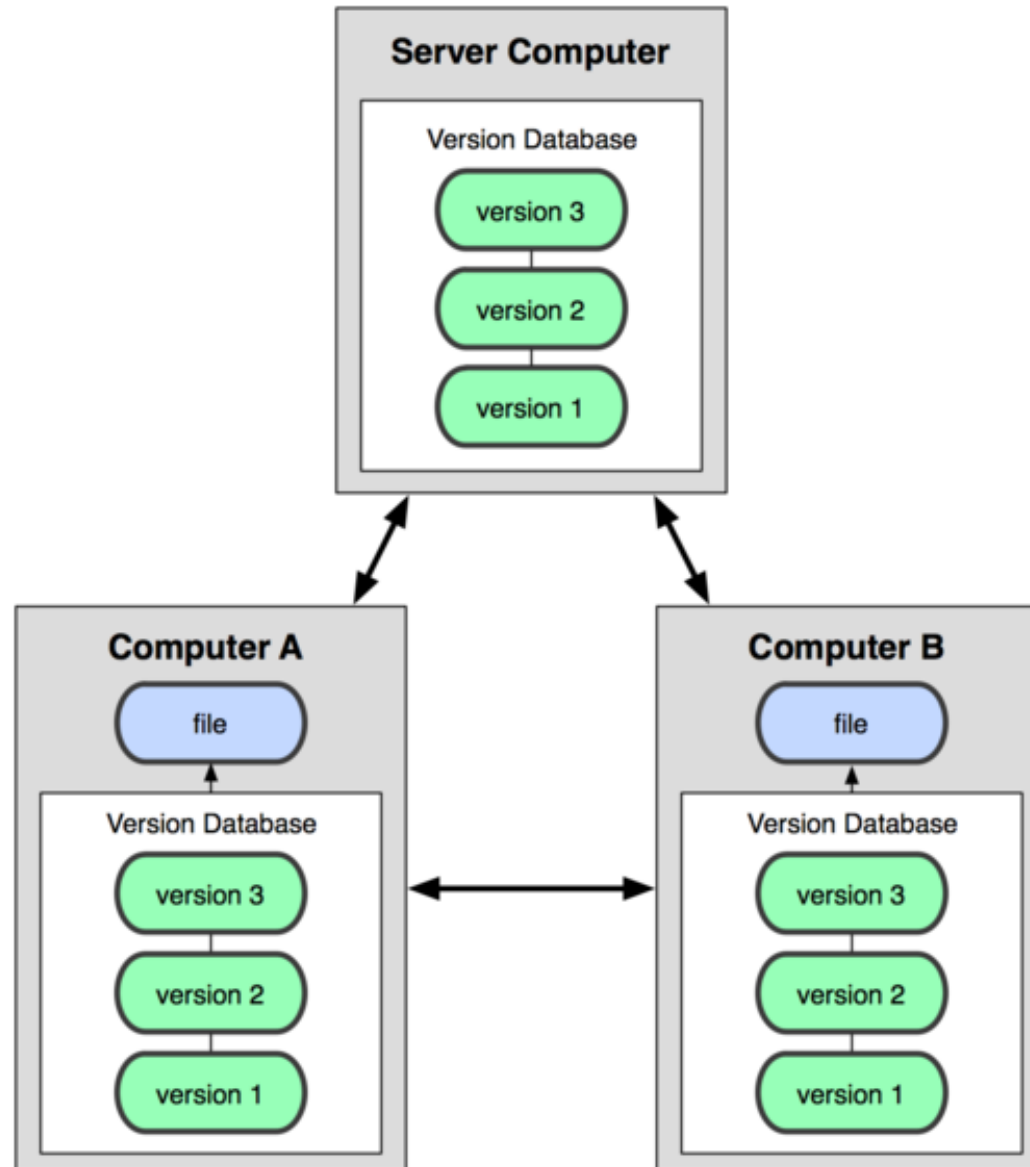


Subversion is like this

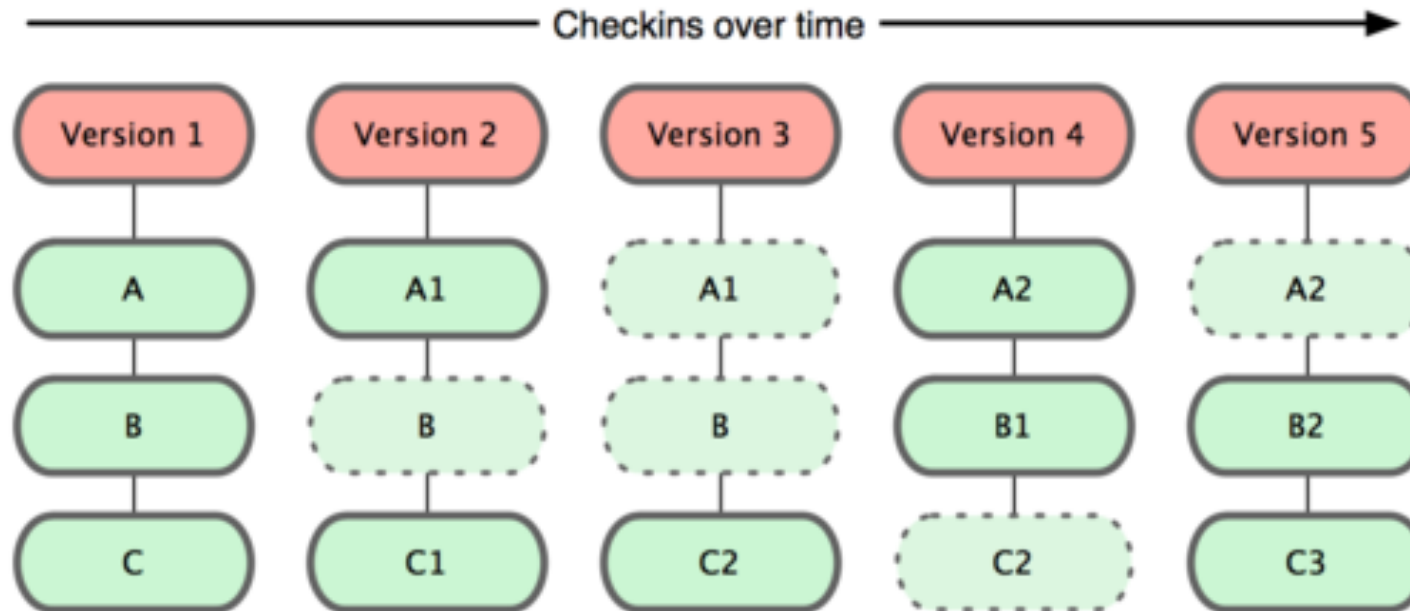
Centralized - Differences



Distributed Version Control

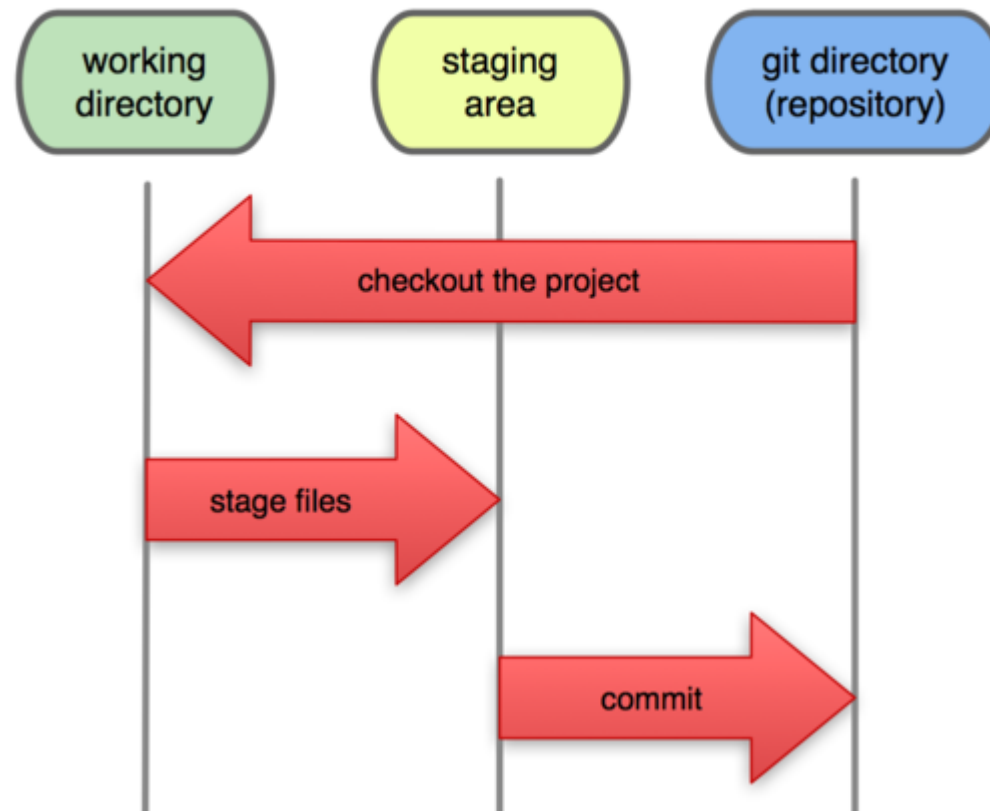


Distributed - Snapshots



- Files are stored by SHA-1 hash rather than filename
- Stored in git database in compressed format
- Database is stored on your *local* machine
- Must “checkout” from database into working directory to edit
- In this example, files A, B and C are *tracked*

Local Operations



Why might you want to stage files?

Tell Git who you are



- Work with Git bash (GUIs also available)
 - cd = change directory (navigate to your files)
 - ls = list the directory contents
- Update your config, one time only
 - git config --global user.name “Cyndi Rader”
 - git config --global user.email crader@mines.edu
 - git config --global core.editor notepad++
 - git config --list

.gitignore



- It's important to tell Git what files you do *not* want to track
- Temp files, executable files, etc. do not need version control (and can cause major issues when merging!)
- <https://help.github.com/articles/ignoring-files>
- Example (place in root of repo):
 - *.class
 - .project
 - .classpath
 - .settings/

Git help



```
Renatos-MacBook-Pro:grit renatonovais$ git help config
```

```
GIT-CONFIG(1)
```

```
Git Manual
```

```
GIT-CONFIG(1)
```

NAME

```
git-config - Get and set repository or global options
```

SYNOPSIS

```
git config [<file-option>] [type] [-z|--null] name [value [value_regex]]
git config [<file-option>] [type] --add name value
git config [<file-option>] [type] --replace-all name value [value_regex]
git config [<file-option>] [type] [-z|--null] --get name [value_regex]
git config [<file-option>] [type] [-z|--null] --get-all name [value_regex]
git config [<file-option>] [type] [-z|--null] --get-regexp name_regex [value_regex]
git config [<file-option>] [type] [-z|--null] --get-urlmatch name URL
git config [<file-option>] --unset name [value_regex]
git config [<file-option>] --unset-all name [value_regex]
git config [<file-option>] --rename-section old_name new_name
git config [<file-option>] --remove-section name
git config [<file-option>] [-z|--null] -l | --list
git config [<file-option>] --get-color name [default]
git config [<file-option>] --get-colorbool name [stdout-is-tty]
git config [<file-option>] -e | --edit
```

DESCRIPTION

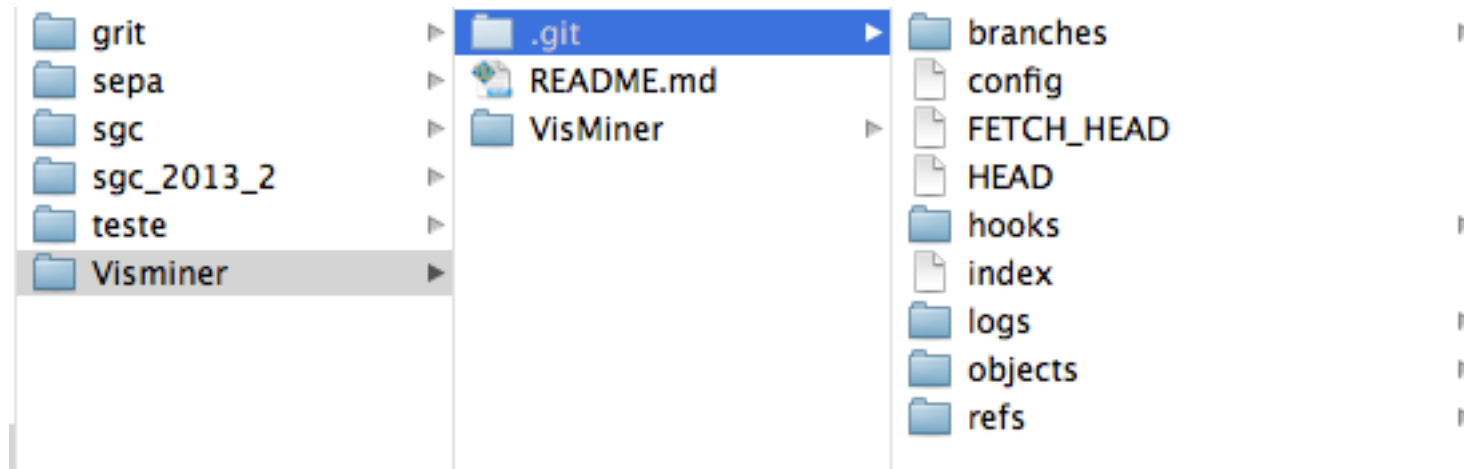
You can query/set/replace/unset options with this command. The name is actually the section and the key separated by a dot, and the value will be escaped.

Multiple lines can be added to an option by using the `--add` option. If you want to update or unset an option which can occur on multiple lines, a POSIX regexp `value_regex` needs to be given. Only the existing values that match the regexp are updated or unset. If you want to

Iniciando um projeto GIT



- Entre num diretório do projeto e digite
 - Git init



Monitorando um arquivo



Nothing tracked yet.

```
$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       HelloGit.java
nothing added to commit but untracked files present (use "git add" to track)
```

Tell Git to track a file

```
$ git add *.java
Cyndi@CYNDI-PC /c/csm_classes/both/cs306/javacode/1
er)
$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   HelloGit.java
#
```

(use "git rm --cached <file>..." to unstage)

Commit the file to the Git database



```
$ git commit -m "Initial project version"
[master (root-commit) d2203a3] Initial project version
1 file changed, 12 insertions(+)
create mode 100644 HelloGit.java

Cyndi@CYNDI-PC /c/csm_classes/ath/cs306/javacode/practice
er)
$ git status
# On branch master
nothing to commit (working directory clean)
```

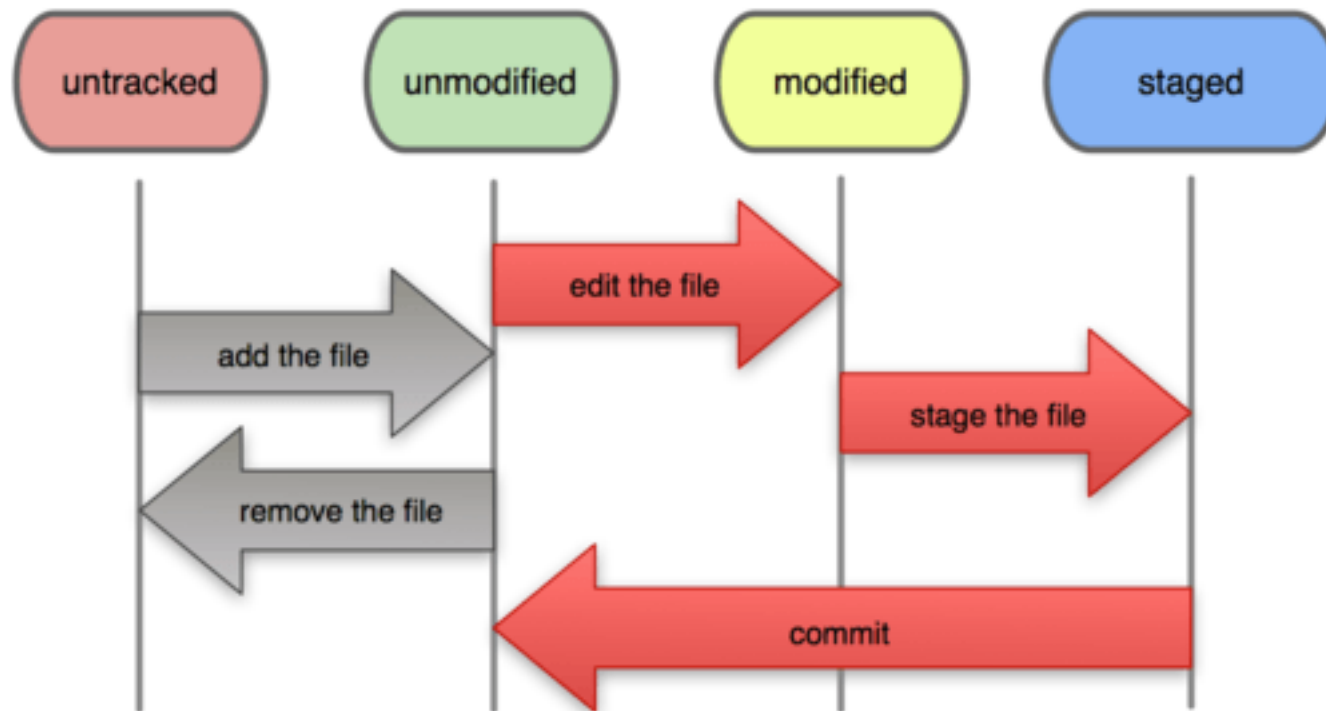
When you commit, you must provide a comment (if you forget, Git will open a text editor so you can write one).

Clonando um repositório remoto



```
git clone git://github.com/visminer/Visminer
```

File Status Lifecycle



Visualizando mudanças



- Git diff => compara Working directory com staged area
- Git diff -staged => compara staged area com último commit

```
$ git diff
diff --git a/benchmarks.rb b/benchmarks.rb
index 3cb747f..da65585 100644
--- a/benchmarks.rb
+++ b/benchmarks.rb
@@ -36,6 +36,10 @@ def main
     @commit.parents[0].parents[0].parents[0]
     end

+   run_code(x, 'commits 1') do
+     git.commits.size
+   end
+
   run_code(x, 'commits 2') do
     log = git.commits('master', 15)
     log.size
```

You made some changes – but what did you do?



```
$ git diff
diff --git a/HelloGit.java b/HelloGit.java
index edeff09..76331ba 100644
--- a/HelloGit.java
+++ b/HelloGit.java
@@ -3,6 +3,7 @@ public class HelloGit {

    public static void main(String[] args) {
        System.out.println("Hello Git!");
+       System.out.println("It's so good to know you.");
    }
}
```

This command compares your working directory with your staging area.
These are the changes that are not yet staged.

Pulando a area de staged



```
git commit -a -m 'comment'
```

```
$ git status
# On branch master
#
# Changes not staged for commit:
#
#   modified:   benchmarks.rb
#
$ git commit -a -m 'added new benchmarks'
[master 83e38c7] added new benchmarks
1 files changed, 5 insertions(+), 0 deletions(-)
```

Removendo arquivo



Removendo da pasta

```
$ rm grit.gemspec
$ git status
# On branch master
#
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#
#       deleted:    grit.gemspec
#
```

Removendo do git

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

So what all have I done?



```
$ git log
commit 575e14397d378402a7ff5b8d44796be5c85fb373
Author: Cyndi Rader <crader@mines.edu>
Date: Thu Aug 29 17:52:22 2013 -0600

    Nice message to Git

commit eceeb75af3e06a48386da7fe77026a35d6d9d31e
Author: Cyndi Rader <crader@mines.edu>
Date: Thu Aug 29 17:43:16 2013 -0600

    Saying hello to Git

commit d2203a3833b0f3a15df66d2cb70b3d740efb0c3b
Author: Cyndi Rader <crader@mines.edu>
Date: Thu Aug 29 17:39:24 2013 -0600

    Initial project version
```

- There are many useful options for git log.

Gitk: um aplicativo visual para log



The screenshot shows the Gitk application window titled "gitk: grit". On the left, a commit log is displayed with a vertical timeline. The selected commit is highlighted in green. The commit message reads: "Clarify how to get a full count out of Repo#comm". The commit details pane on the right shows the commit hash "2a8c38b72380fc64700d0e8e6d7b16d45dfb6ebd", the author "Jos Backus <jos@catnook.com>" with a timestamp of "2009-01-30 17:49:40", and the committer "Scott Chacon <schacon@gmail.com>" with a timestamp of "2009-01-30 18:23:08". The commit message is repeated in the main pane, followed by a diff for the file "lib/grit/git.rb". The diff shows a change in the "include GitRuby" line and the addition of a new attribute accessor for "git_max_size".

```
----- lib/grit/git.rb -----
index 7c1785d..fe3d6b5 100644
@@ -22,11 +22,19 @@ module Grit
   include GitRuby

   class << self
-     attr_accessor :git_binary, :git_timeout
+     attr_accessor :git_binary, :git_timeout, :git_max_size
   end

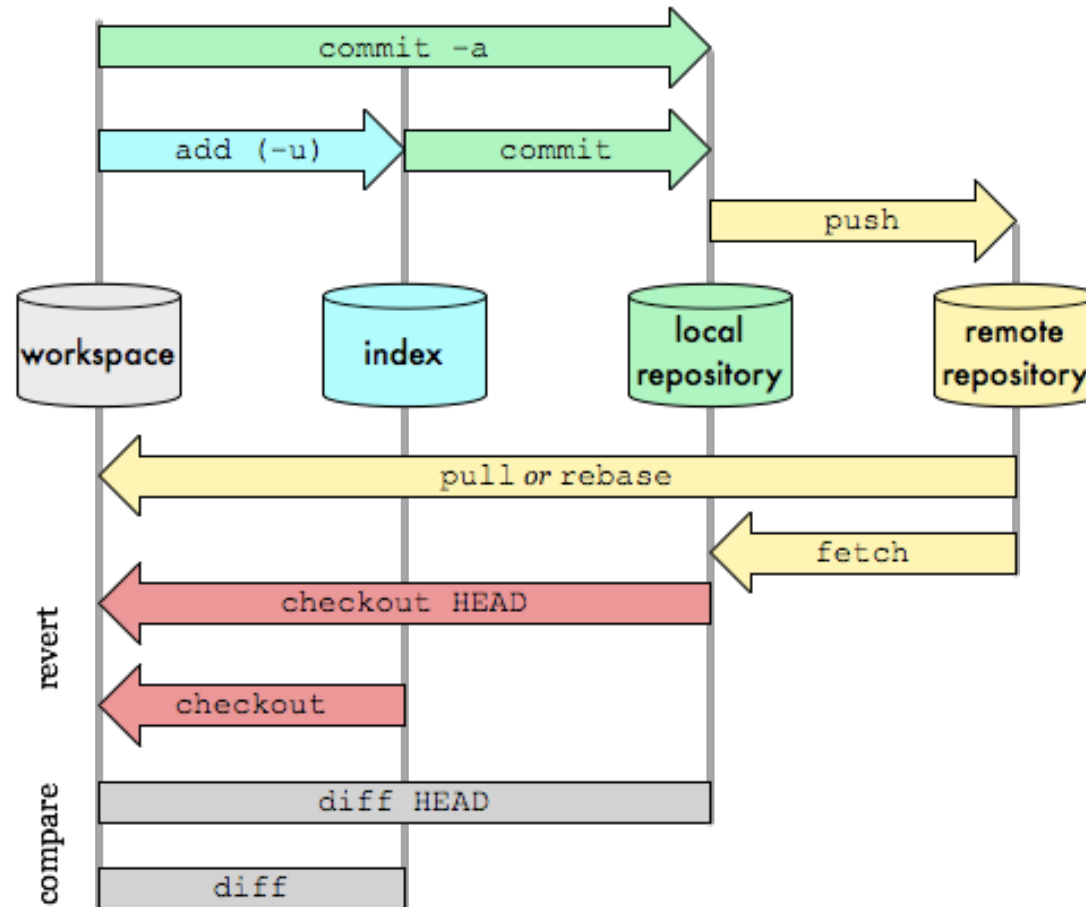
-   self.git_binary = "/usr/bin/env git"
```


The Big Picture



Git Data Transport Commands

<http://osteele.com>



<http://blog.mikepearce.net/2010/05/18/the-difference-between-git-pull-git-fetch-and-git-clone-and-git-rebase/>

- Cuidado!! Poucas coisas no git não podem ser desfeitas, essa é uma delas. Você pode perder dados se fizer errado.
- Modificando o último commit
 - Depois desses três comandos você obterá um único commit — o segundo commit substitui os resultados do primeiro.

```
$ git commit -m 'initial commit'  
$ git add forgotten_file  
$ git commit -amend -m 'nova  
mensagem'
```

Desfazendo coisas no git



- Tirando um arquivo da área de seleção
- Imagine que você colocou dois arquivos na área staged e quer fazer commits diferentes para eles.
- O próprio comando **git status** ajuda nisso

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

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

Desfazendo um arquivo modificado



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

- **Cuidado com isso!!!**

```
$ git checkout -- benchmarks.rb
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   README.txt
#
```

- A colaboração acontece efetivamente através dos repositórios remotos
- Podem ter vários repositórios remotos
 - Uns só leitura, outros leitura/escrita, etc
- Colaborar envolve gerenciar esses repositórios
 - Fazer push e pull de dados
- Atividades
 - Adicionar e remover repositórios remotos
 - Gerenciar branches e defini-los como monitorados ou não

Exibindo os repositórios remotos



- git remote
 - Permite ver os repositórios remotos, se você tiver clonado pelo menos um chamado origin aparece

```
$ git clone git://github.com/schacon/ticgit.git
Initialized empty Git repository in /private/tmp/ticgit/.git/
remote: Counting objects: 595, done.
remote: Compressing objects: 100% (269/269), done.
remote: Total 595 (delta 255), reused 589 (delta 253)
Receiving objects: 100% (595/595), 73.31 KiB | 1 KiB/s, done.
Resolving deltas: 100% (255/255), done.
```

```
$ cd ticgit
$ git remote
origin
```

- git remote -v
 - Mostra url completas

Adicionando repositórios remotos



- Git remote add [alias] [url]

```
$ git remote  
origin  
$ git remote add pb git://github.com/paulboone/  
ticgit.git  
$ git remote -v  
origin git://github.com/schacon/ticgit.git  
pb git://github.com/paulboone/ticgit.git
```

Pegando dados de repositórios remotos



- Git fetch [nome-repositorio]
 - Traz os dados do repositório remoto para o local
- Se você fizer sobre o repositório origem, ele trará os dados modificados a partir do momento que você clonou o repositório
- Se você adicionou o repositório, você deve fazer o fetch para pegar os dados
- Não faz o merge dos dados

```
$ git fetch pb
remote: Counting objects: 58, done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 44 (delta 24), reused 1 (delta 0)
Unpacking objects: 100% (44/44), done.
From git://github.com/paulboone/ticgit
* [new branch]      master    -> pb/master
* [new branch]      ticgit    -> pb/ticgit
```


- Git pull
 - Traz os dados do repositório remoto para o branch (working directory) que foi configurado no seu repositório local (faz o fetch e o merge) para poder acompanhar o repositório remoto.
 - Isso acontece automaticamente com o origin. O git faz uma associação entre o branch master desse repositório e o master do remoto (se houver)

Enviando dados para repositório remoto



- Após o commit no repositório local, realize o comando
 - `git push [nome branch local] [nome branch remoto]`

Inspecionando um repositório remoto



- Git remote show origin

```
$ git remote show origin
* remote origin
  URL: git://github.com/schacon/ticgit.git
  Remote branch merged with 'git pull' while on branch master
  master
  Tracked remote branches
  master
  ticgit
  Local branch pushed with 'git push'
  master:master
```

Branch **remoto**
usado no merge
com o git pull

Branch **local**
usado no merge
com o git pull

Renomeando e Removendo repositórios remotos



- Git remote rename pb pall
 - Serve para renomear um repositório

```
$ git remote rename pb paul
$ git remote
origin
paul
```

- Git remote rm paul

```
$ git remote rm paul
$ git remote
origin
```

- Listando tags
 - git tag
- Listando tags com um nome particular
 - git tag -l 'v*'
- Tipos de tags
 - Leve: similar a um branch que não muda, é um ponteiro para um commit específico
 - Anotada, é um objeto completo no banco de dados git

Criando tags anotadas

```
$ git tag -a v1.4 -m 'my version 1.4'  
$ git tag  
v0.1  
v1.3  
v1.4
```

Criando tags leves

```
$ git tag v1.4-lw  
$ git tag  
v0.1  
v1.3  
v1.4  
v1.4-lw  
v1.5
```

Criando Branchs



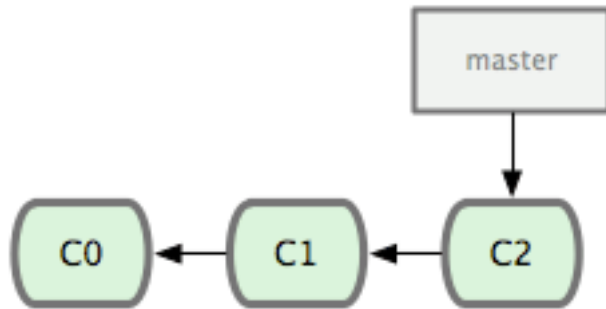
- Criar um branch: `git branch <nome do branch>`
 - `git branch iss53`
- Mudar para o branch: `git checkout <nome do branch>`
 - `git checkout iss53`
- Fazendo os dois ao mesmo tempo
 - `git checkout -b iss53`

Cenário de trabalho



- A aplicação está no master
 - Tem um arquivo index.html
- Você precisa fazer uma correção de um bug #53
 - Criar um branch iss53 para corrigir o bug
 - Atualizar o arquivo index.html
- Você é avisado que precisa urgentemente resolver um problema (hotfix) no master
 - Criar um branch hotfix para resolver esse problema
 - Atualizar o arquivo index.html (em linhas diferentes – evitar conflito agora)
- Passar o que foi feito em hotfix para o master

Branch Master

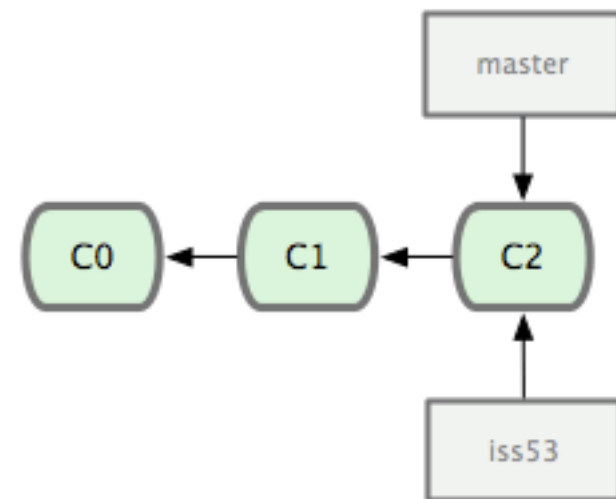


- Faça o master ter um arquivo index.html. Crie, add e commit.

- Criar o branch iss53 e mudar para ele

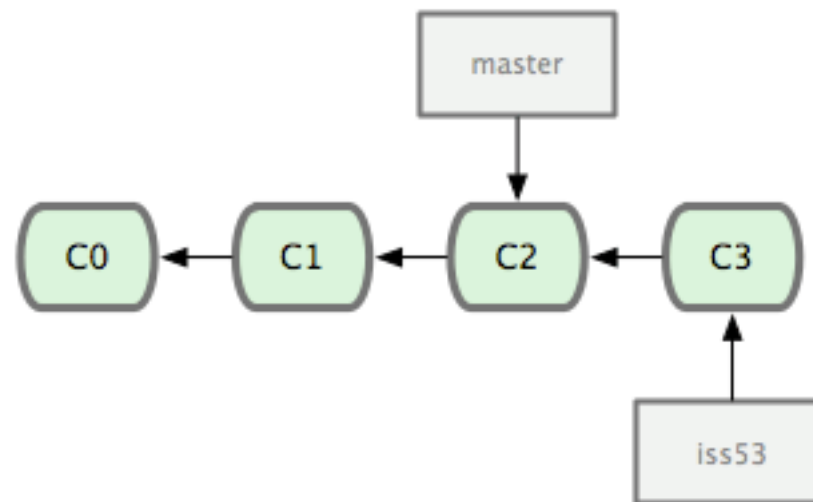
```
$ git checkout -b iss53  
Switched to a new branch "iss53"
```

```
$ git branch iss53  
$ git checkout iss53
```



- Edite o arquivo index.html, e commit
- Use o gitk para acompanhar

```
$ vim index.html  
$ git commit -a -m 'adicionei um novo rodapé  
[issue 53]'
```



Solicitação da mudança hotfix

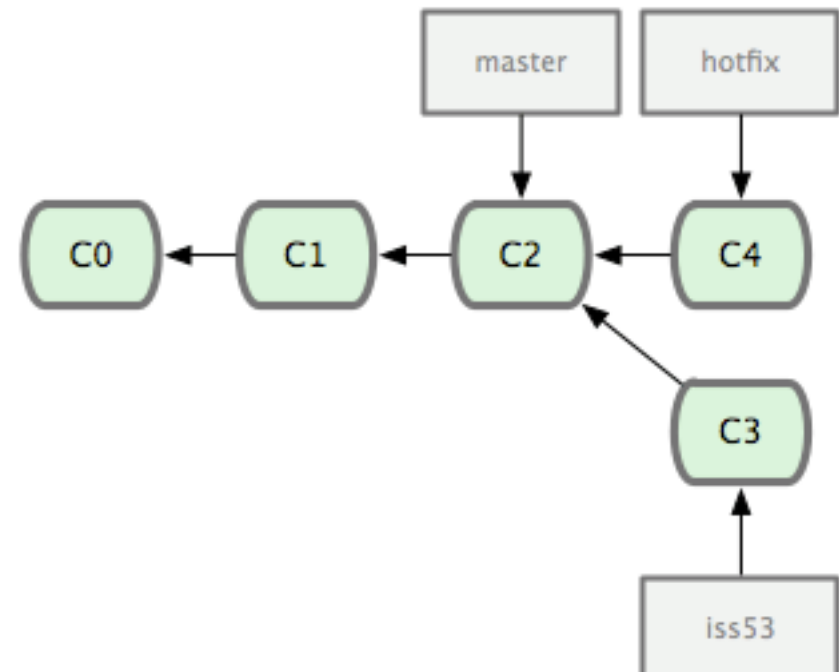


- Volte para o master, pois seu iss53 ainda está incompleto

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

- Crie o branch hotfix
- Edite o arquivo index.html, e commit

```
$ git checkout -b 'hotfix'  
Switched to a new branch "hotfix"  
$ vim index.html  
$ git commit -a -m 'concertei o endereço de email'  
[hotfix]: created 3a0874c: "concertei o endereço de email"  
1 files changed, 0 insertions(+), 1 deletions(-)
```

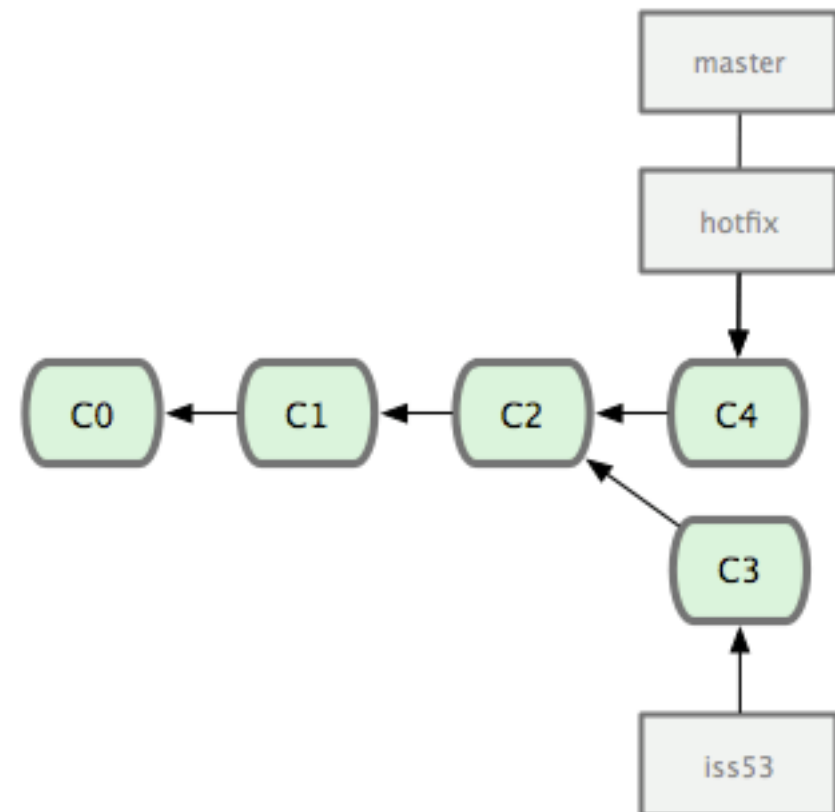


Agora fazer o merge



- Vá para o branch master
- E solicite para fazer o merge do hotfix
- Fast forward: o git percebeu que o commit do branch master estava antes do commit do branch do merge (hotfix), como não tem conflito, ele só faz avançar o ponteiro (fast forward)

```
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast forward
 README | 1 -
 1 files changed, 0 insertions(+), 1
 deletions(-)
```



Delete branch

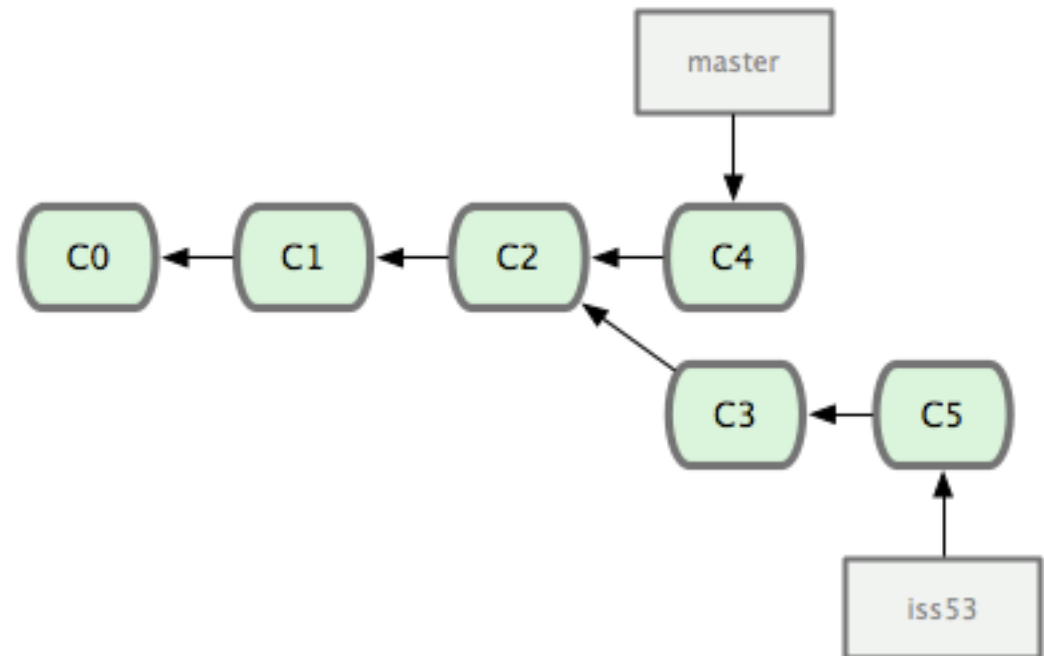


- Delete o branch hotfix que você não precisa mais dele

```
$ git branch -d hotfix  
Deleted branch hotfix (3a0874c).
```

- Volte a trabalhar agora no branch iss53

```
$ git checkout iss53  
Switched to branch "iss53"  
$ vim index.html  
$ git commit -a -m 'novo rodapé terminado  
[issue 53]'  
[iss53]: created ad82d7a: "novo rodapé  
terminado [issue 53]"  
1 files changed, 1 insertions(+), 0  
deletions(-)
```

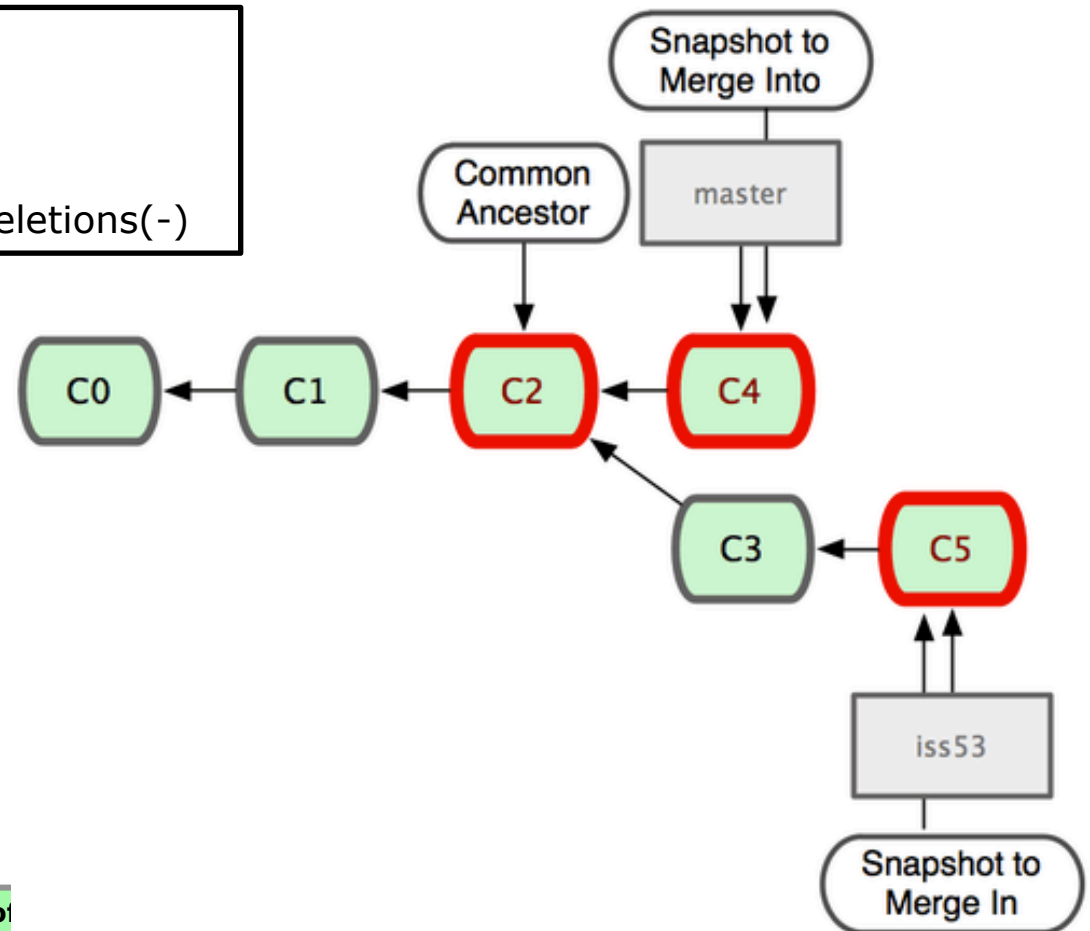


Fazer o merge do iss53

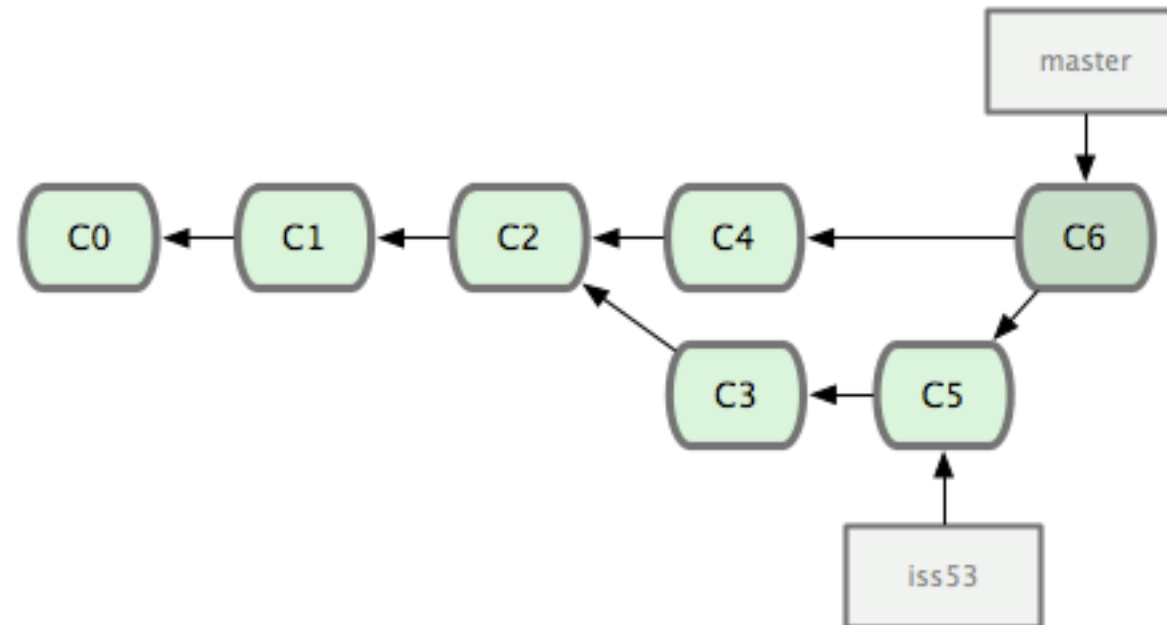
- Observe que agora não pode haver mais o fast forward, eles evoluíram em caminhos diferentes.
- O histórico dos commits divergem

```
$ git checkout master  
$ git merge iss53  
Merge made by recursive.  
README | 1 +  
1 files changed, 1 insertions(+), 0 deletions(-)
```

- O git usa os três snapshots para fazer o merge



- Resultado final



- Delete o branch iss53
- `git branch -d iss53`

Gerenciando conflito



- Se foi alterado a mesma parte do arquivo, o processo vai gerar conflito
- Resolução manual

```
$ git merge iss53
Auto-merging index.html
CONFLICT (content): Merge conflict in
index.html
Automatic merge failed; fix conflicts and then
commit the result.
```

- O git não gera o commit automaticamente, ele pausa o processo
- Para ver quais arquivos deram conflito, faça:

```
[master*]$ git status
index.html: needs merge
# 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)
#
#   unmerged:   index.html
#
```

Abra o arquivo



- Git adiciona marcadores padrão no arquivo

```
<<<<<<< HEAD:index.html
<div id="footer">contato : email.support@github.com</div>
=====
<div id="footer">
  por favor nos contate em support@github.com
</div>
>>>>>>> iss53:index.html
```

- Tudo que tem acima de ===== é do branch master, tudo que tem abaixo é do branch que você quer fazer o merge (iss53)
- Modifica o arquivo manualmente e faz o add e commit no master

Ferramenta visual para gerenciar o conflito



- git mergetool

```
$ git mergetool
merge tool candidates: kdiff3 tkdiff xxdiff meld gvimdiff opendiff emerge vimdiff
Merging the files: index.html

Normal merge conflict for 'index.html':
 {local}: modified
 {remote}: modified
Hit return to start merge resolution tool (opendiff):
```

Mais sobre branches



- Git branch (mostra todos os branches), o * aponta para o branch atual (que fez o checkout)

```
$ git branch
iss53
* master
testing
```

- Para ver o ultimo commit em cada branch

```
$ git branch -v
iss53 93b412c concertar problema em javascript
* master 7a98805 Merge branch 'iss53'
testing 782fd34 adicionar scott para a lista de autores nos readmes
```

- Mostrar apenas branches com merge ou os sem merge

```
$ git branch --merged
iss53
* master
```

```
$ git branch --no-merged
testing
```

- Update your config, one time only
 - `git config --global user.name "Cyndi Rader"`
 - `git config --global user.email crader@mines.edu`
 - `git config --list`
- Crie uma pasta do projeto e realize o Git init
- Git status
- Git add
- Git commit
- Git diff

Perguntas



?

Referências



- Material compilado de
 - <http://git-scm.com/book>