Introdução ao Git

Metodologias e Desenvolvimento de Software

Pedro Salgueiro
pds@di.uevora.pt
CLV-256

Instalação

- Instalar em Linux (sistemas baseados em debian)
  - Debian, Ubuntu, Linux Mint, etc..

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

- Instalar no Windows
  - `http://git-scm.com/download/win`

Configuração

- Ferramenta de configuração
  - ler e especificar variáveis de configuração
  - `git config`
    - System config
      - `/etc/gitconfig`
    - User config
      - Linux: `~/.gitconfig` or `~/.config/git/config`
      - Windows:
    - Repositoty
      - .git/config
- Identidade pessoal

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

Configuração

- Identidade

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

- Editor de texto

```
$ git config --global core.editor emacs
```

Configuração

- Verificar a configuração (global)

```
$ git config --list
user.name=John Doe
user.email=johndoe@example.com
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
...
```

- Verificar configurações especificas
  - `git config <key>`

```
$ git config user.name
```

Obter um repositório

- Iniciar um repositório numa directoria existente
- Clonar um repositório existente

Obter um repositório

- Inicializar um repositório numa directoria existente

```
$ git init
```

  - Começar a “monitorizar” (tracking)
    - git add
    - git commit

```
$ git add *.c
$ git add LICENSE
$ git commit –m 'initial project version'
```

Obter um repositório

- Clonar um repositório existente
  - `git clone [url] [directory]`
  - `url`: http, https, git, ssh

```
$ git clone https://github.com/libgit2/libgit2
```

```
$ git clone https://github.com/libgit2/libgit2 mylibgit
```

```
$ git clone user@server:path/to/repo.git
```

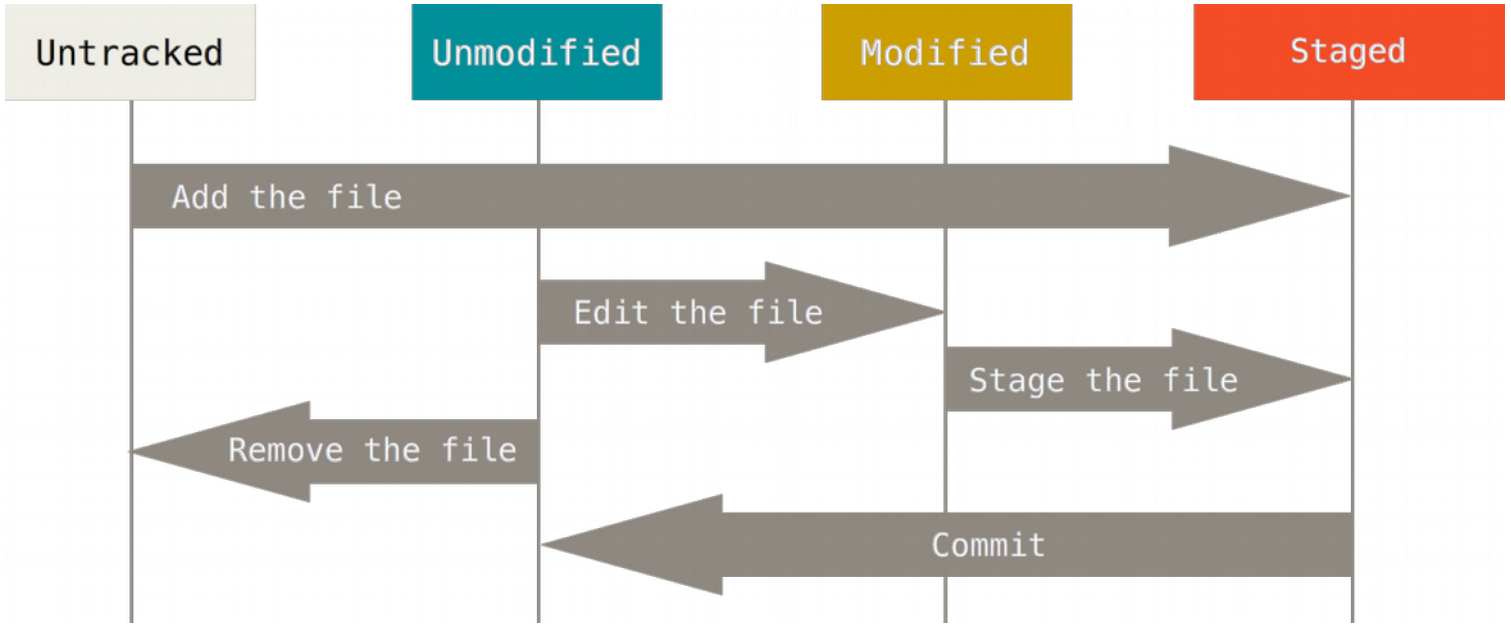
Registar alterações no repositório

- “guardar alterações” no repositório
- Fazer “commit” das alterações
- Estado dos ficheiros
  - untracked
  - tracked
    - unmodified
    - modified
    - staged

Registar alterações no repositório

- Clonar um repositório
  - Todos os ficheiros estão tracked e não modificados
  - Editar um ficheiro
    - modified
    - staged
    - commit

Registar alterações no repositório



Registar alterações no repositório

- Verificar o estado dos ficheiros
  - `git status`

```
$ git status
On branch master
nothing to commit, working directory clean
```

    - Sem alterações e sem ficheiros novos

Registar alterações no repositório

- Verificar o estado dos ficheiros
  - Depois de adicionar um ficheiro

```
$ echo 'My Project' >  README
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be
committed)

    README

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

Registar alterações no repositório

- Monitorizar (fazer o tracking) de novos ficheiros
  - stage do ficheiro
  - `git add README`

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

    new file:   README
```

Registar alterações no repositório

- “Staging” de ficheiros modificados
  - `CONTRIBUTING.md` alterado

```
$ 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&gt;..." to update what will be
committed)
  (use "git checkout -- <file>..." to discard changes in
working directory)

    modified:   CONTRIBUTING.md
```

Registar alterações no repositório

- “Staging” de ficheiros modificados
  - `git add CONTRIBUTING.md`

```
$ git add CONTRIBUTING.md
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   README
    modified:   CONTRIBUTING.md
```

  - Ambos os ficheiros estão incluidos no próximo commit

Registar alterações no repositório

- “Staging” de ficheiros modificados
  - `CONTRIBUTING.md` alterado (novamente)

```
$ vim CONTRIBUTING.md
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   README
    modified:   CONTRIBUTING.md

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:   CONTRIBUTING.md
```

Registar alterações no repositório

- “Staging” de ficheiros modificados
  - `CONTRIBUTING.md` alterado (novamente)
    - **staged** para commit
    - **not staged** para commit
    - ao mesmo tempo
    - porquê?
  - Git “faz stage” do ficheiro exactamente no estado actual:
    - `git add`
  - O que vai ser “commitado”?
    - O ficheiro no estado em que estava quando foi adicionado/staged

Registar alterações no repositório

- “Staging” de ficheiros modificados
  - `git add CONTRIBUTING.md` (novamente)

```
$ git add CONTRIBUTING.md
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   README
    modified:   CONTRIBUTING.md
```

  - `CONTRIBUTING.md` alterado (novamente)
    - **staged** para commit (only)

Registar alterações no repositório

- “Staging” de ficheiros modificados

  - Estado do repositório
    - Versão *resumida*
      - `git status -s`

```
$ git status -s
 M README
MM Rakefile
A  lib/git.rb
M  lib/simplegit.rb
?? LICENSE.txt
```

- A → staged for commit
- M → modified
- ?? → new/untracked files

Registar alterações no repositório

- Ignorar ficheiros
  - Ficheiros que não devem ser monitorizados pelo git
  - .gitignore
    - Especificados através de padrões de nomes
      - Linhas em branco ou que comecem por # → ignoradas
      - Standard glob expressions
        - Expressões regulares usadas para especificar nomes de ficheiros
        - Exemplos: *.txt, cliente.*, ???.txt, etc
      - Começar com um "/" para evitar recursividade dos nomes
      - Terminar com "/" para especificar uma directoria
      - Simplified regular expressions
        - * → 0 ou mais caracteres
        - [abc] → qualquer um dos caracteres especificados
        - ? → um unico caracter
        - [0-9] → qualquer caracter entre 0 e 9
        - ** → directorias imbricadas
          - a/**/z → a/z, a/b/z, a/b/c/z, etc…

Registar alterações no repositório

- Ignorar ficheiros (exemplo)

```
# no .a files
*.a

# but do track lib.a, even though you're ignoring .a files above
!lib.a

# only ignore the TODO file in the current directory, not
subdir/TODO
/TODO

# ignore all files in the build/ directory
build/

# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt

# ignore all .pdf files in the doc/ directory
doc/**/*.pdf
```

Registar alterações no repositório

- Ver alterações
  - Ainda não marcadas para *stage*
    - `git diff`

```
$ git diff
diff --git a/CONTRIBUTING.md b/CONTRIBUTING.md
index 8ebb991..643e24f 100644
--- a/CONTRIBUTING.md
+++ b/CONTRIBUTING.md
@@ -65,7 +65,8 @@ branch directly, things can get messy.
 Please include a nice description of your changes when you submit your PR;
 if we have to read the whole diff to figure out why you're contributing
 in the first place, you're less likely to get feedback and have your change
-merged in.
+merged in. Also, split your changes into comprehensive chunks if your patch
is
+longer than a dozen lines.

 If you are starting to work on a particular area, feel free to submit a PR
 that highlights your work in progress (and note in the PR title that it's
```

Registar alterações no repositório

- Fazer commit das alterações
  - `git commit`
    - Abre o editor de texto default
    - `git config --global core.editor`

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#   new file:   README
#   modified:   CONTRIBUTING.md
#
~
~
~
".git/COMMIT_EDITMSG" 9L, 283C
```

Registar alterações no repositório

- Fazer commit das alterações
  - Método alternativo
    - `git commit –m “some commit message”`

```
$ git commit –m "Story 182: Fix benchmarks for speed"
[master 463dc4f] Story 182: Fix benchmarks for speed
 2 files changed, 2 insertions(+)
 create mode 100644 README
```

Registar alterações no repositório

- Fazer commit das alterações
  - Evitar a área de staging
    - `git commit -a -m "some commit message"`

```
$ git commit -a -m 'added new benchmarks'
[master 83e38c7] added new benchmarks
 1 file changed, 5 insertions(+), 0 deletions(-)
```

Registar alterações no repositório

- Apagar ficheiros
  - Apagar o ficheiro não é suficiente
    - unstaged

```
$ rm PROJECTS.md
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working
directory)

        deleted:    PROJECTS.md

no changes added to commit (use "git add" and/or "git commit -a")
```

Registar alterações no repositório

- Apagar ficheiros
  - `git rm filename`
    - Apaga, remove dos *tracked files*

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

    deleted:    PROJECTS.md
```

Registar alterações no repositório

- Apagar ficheiros apenas do repositório
  - Fazer “untrack” de um ficheiro
  - Remover o ficheiro da area de “stage”
    - Mas mantê-lo na directoria de trabalho
  - Util quando nos esquecemos de adicionar um ficheiro ao .gitignore

```
$ git rm --cached README
```

Registar alterações no repositório

- Mover ficheiros
  - `git mv file_from file_to`

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

    renamed:    README.md -> README
```

Registar alterações no repositório

- Mover ficheiros
  - Em alternativa

```
$ mv README.md README
$ git rm README.md
$ git add README
```

Consultar o histórico de commits

```
$ git log
commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the version number

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 16:40:33 2008 -0700

    removed unnecessary test

commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 10:31:28 2008 -0700

    first commit
```

Desfazer “coisas”

- Nem sempre é possível desfazer coisas
- Corrigir o ultimo commit
  - Ficheiros esquecidos ou mensagem de commit errada
  - `git commit --amend`

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

Desfazer “coisas”

- Fazer *unstage* de um ficheiro
  - Ficheiro foi adicionado acidentalmente

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

    renamed:    README.md -> README
    modified:   CONTRIBUTING.md
```

Desfazer coisas

- Fazer *unstage* de um ficheiro
  - manter as alterações na directoria local
  - `git reset HEAD CONTRIBUTING.md`

```
$ git reset HEAD CONTRIBUTING.md
Unstaged changes after reset:
M   CONTRIBUTING.md
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    renamed:    README.md -> 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:   CONTRIBUTING.md
```

Desfazer coisas

- Fazer “undo” a um ficheiro alterado
  - “Esquecer as alterações a um ficheiro”
  - `git checkout -- filename`

```
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:   CONTRIBUTING.md


$ git checkout -- CONTRIBUTING.md
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    renamed:    README.md -> README
```

Trabalhar com *Remotes*

- Repositórios remotos
  - Versões do projecto, alojadas noutro local
  - Essencial para colaborar num projecto
    - **push**
      - Enviar dados para o repositório
    - **pull**
      - Descarregar dados do repositório

Trabalhar com *Remotes*

- Listar os *remotes*
  - `git remote`

```
$ git clone https://github.com/schacon/ticgit
Cloning into 'ticgit'...
remote: Reusing existing pack: 1857, done.
remote: Total 1857 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (1857/1857), 374.35 KiB | 268.00 KiB/s, done.
Resolving deltas: 100% (772/772), done.
Checking connectivity... done.
$ cd ticgit
$ git remote
origin
```

```
$ git remote -v
origin  https://github.com/schacon/ticgit (fetch)
origin  https://github.com/schacon/ticgit (push)
```

Trabalhar com *Remotes*

- Listar os *remotes*
  - Vários remotes

```
$ cd grit
$ git remote -v
bakkdoor  https://github.com/bakkdoor/grit (fetch)
bakkdoor  https://github.com/bakkdoor/grit (push)
cho45     https://github.com/cho45/grit (fetch)
cho45     https://github.com/cho45/grit (push)
defunkt   https://github.com/defunkt/grit (fetch)
defunkt   https://github.com/defunkt/grit (push)
koke      git://github.com/koke/grit.git (fetch)
koke      git://github.com/koke/grit.git (push)
origin    git@github.com:mojombo/grit.git (fetch)
origin    git@github.com:mojombo/grit.git (push)
```

Trabalhar com *Remotes*

- Adicionar um *remote*
  - `git remote add <shortname> <url>`

```
$ git remote
origin
$ git remote add pb https://github.com/paulboone/ticgit
$ git remote -v
origin  https://github.com/schacon/ticgit (fetch)
origin  https://github.com/schacon/ticgit (push)
pb  https://github.com/paulboone/ticgit (fetch)
pb  https://github.com/paulboone/ticgit (push)
```

Trabalhar com *Remotes*

- *Fetching* e *Pulling* do remote
  - `git fetch [remote-name]`
    - Descarrega todos os dados do projecto a partir de um *remote*
    - Quando se faz clone de um projecto
      - Faz “Fetch” do projecto
      - Adiciona um remoto chamado “origin”
    - Não junta (merge) os dados remotos com os locais
      - Tem que ser feito manualmente
  - `git pull`
    - `fetch`
    - `merge`

Trabalhar com *Remotes*

- *Fetching* e *Pulling*

```
$ git fetch pb
remote: Counting objects: 43, done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 43 (delta 10), reused 31 (delta 5)
Unpacking objects: 100% (43/43), done.
From https://github.com/paulboone/ticgit
 * [new branch]      master     -> pb/master
 * [new branch]      ticgit     -> pb/ticgit
```

Trabalhar com *Remotes*

- Enviar dados para o *remote*
  - *Pushing (empurrar)*
  - Partilhar alterações com outros
  - `git push [remote-name] [branch-name]`
    - Permissões de escrita
    - Não existirem “alterações novas”
      - Ninguém fez push
      - As alterações serão rejeitadas
      - É necessário fazer pull(puxar) as novas alterações, e fazer novo push(empurrar)

```
$ git push origin master
```

Trabalhar com *Remotes*

- Remover e renomear *remotes*

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

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

Tags

- Colocar *tags* em pontos específicos na história do projecto
  - Versões
  - Pontos onde houve *Releases*

- Listar tags

```
$ git tag
v0.1
v1.3
```

```
$ git tag -l "v1.8.5*"
v1.8.5
v1.8.5-rc0
v1.8.5-rc1
v1.8.5-rc2
```

Tags

- Criar tags
  - “apontador” para um commit específico
  - colocar uma tag no commit actual

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

Tags

- Criar tags
  - Criar uma tag depois do commit feito

```
$ git log --pretty=oneline
1502795795 1b64cf874c3557a0f3547bd83b3ff6 Merge branch 'experiment'
a6b4c97498bd301d84096da251c98a07c7723e65 beginning write support
0d52aaab4479697da7686c15f77a3d64d9165190 one more thing
```

```
$ git tag -a v1.2 0d52aaab
```

Tags

- Partilhar tags
  - Por default, `git push` não transfere as tags para os servidores remotos
  - `git push origin [tagname]`

```
$ git push origin v1.5
Counting objects: 14, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (12/12), done.
Writing objects: 100% (14/14), 2.05 KiB | 0 bytes/s, done.
Total 14 (delta 3), reused 0 (delta 0)
To git@github.com:schacon/simplegit.git
 * [new tag]         v1.5 -> v1.5
```

```
$ git push origin --tags
Counting objects: 1, done.
Writing objects: 100% (1/1), 160 bytes | 0 bytes/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To git@github.com:schacon/simplegit.git
 * [new tag]         v1.4 -> v1.4
 * [new tag]         v1.4-lw -> v1.4-lw
```

Tags

- Fazer “checkout” de tags
  - `git checkout -b branchname tagname`
    - Cria um “branch” novo baseado numa tag específica

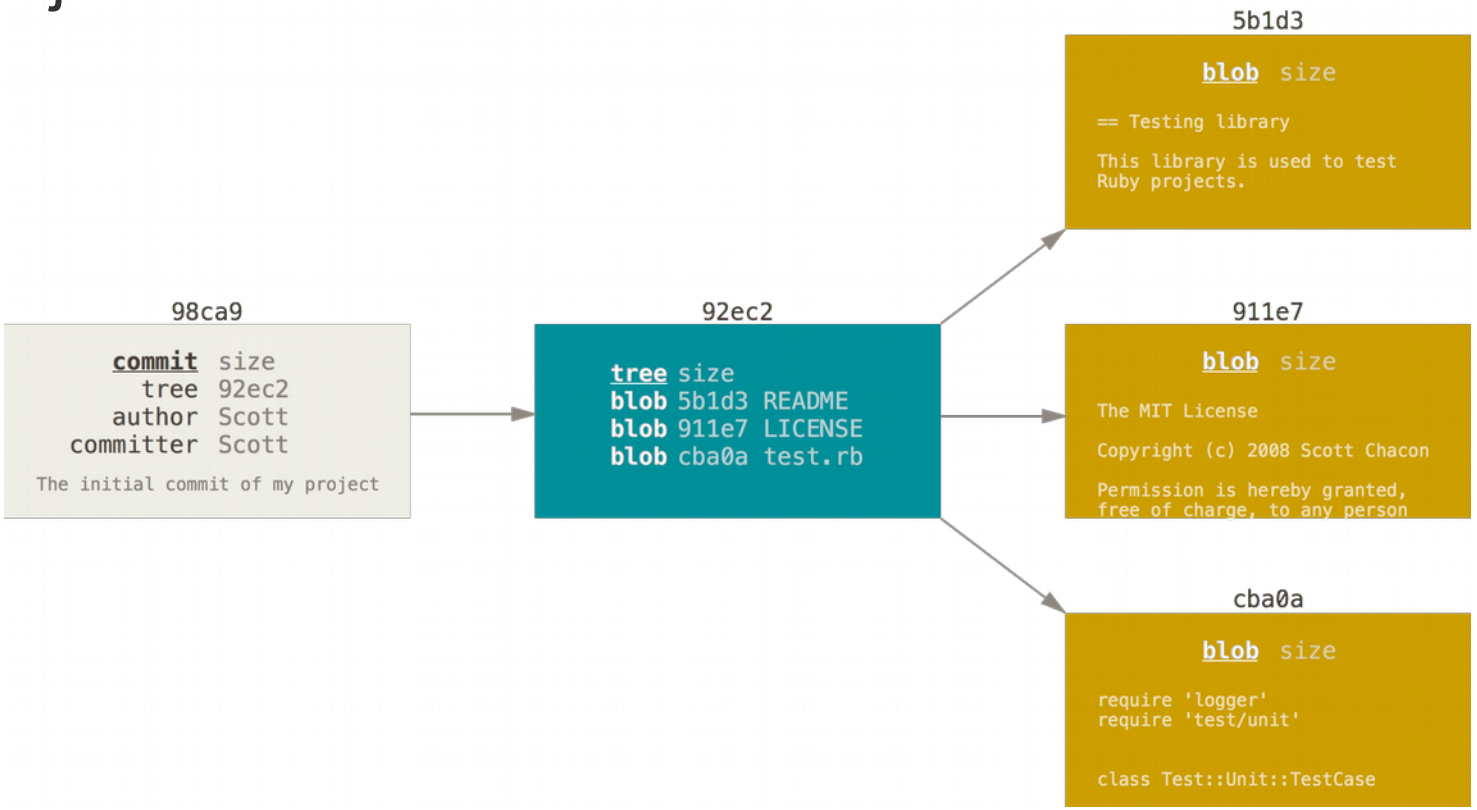
```
$ git checkout -b version2 v2.0.0
Switched to a new branch 'version2'
```

Branches

- Cada commit
  - Objecto de commit
    - Apontador para um snapshot do conteudo que foi “staged”
    - Apontador para os commits anteriores
    - metadados

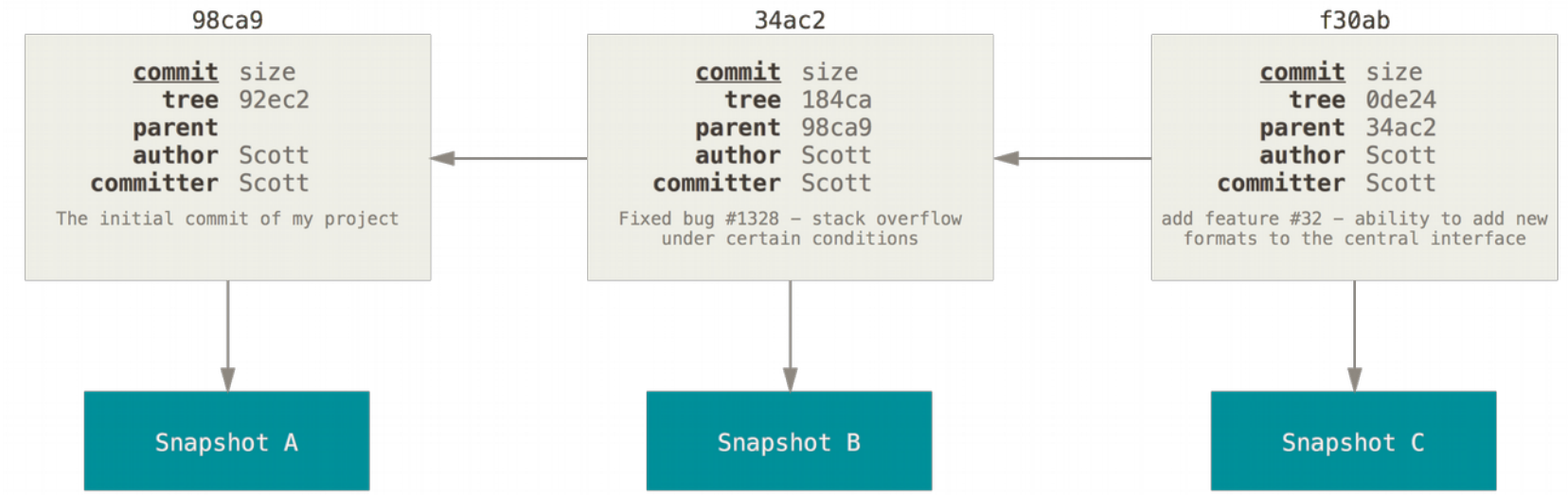
Branches

- Objectos de commit



Branches

- Commit novo
  - Apontador para o commit anterior commit

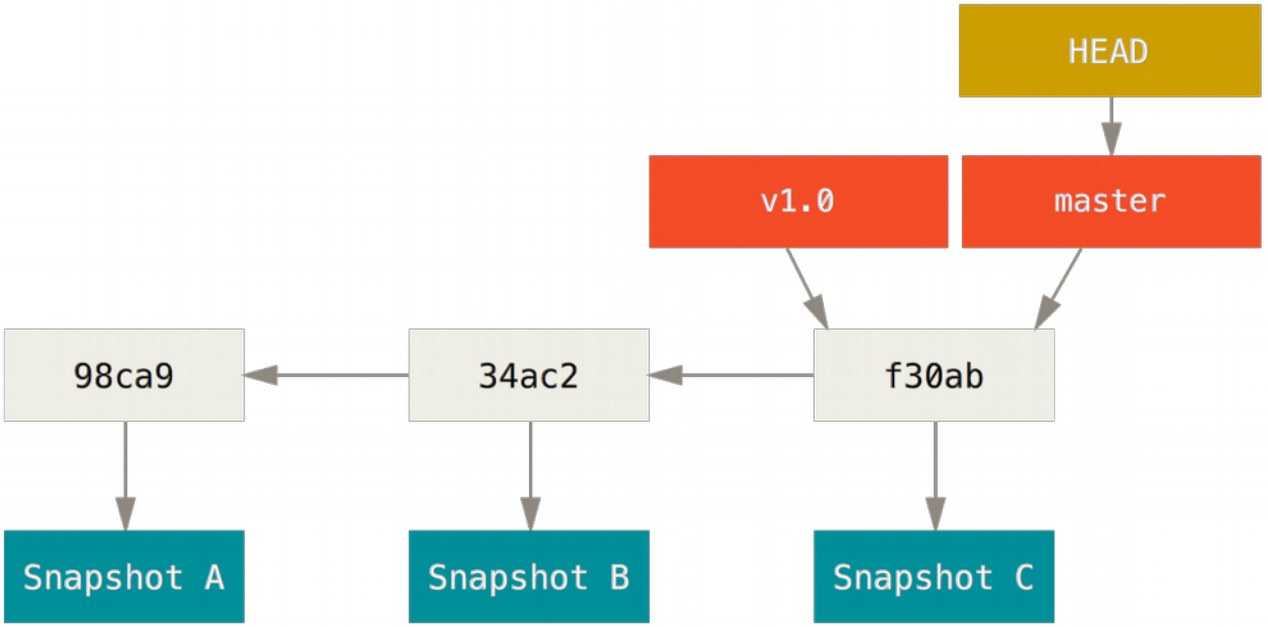


Branches

- Branch
  - Apontador móvel para um commit
  - Branch default: master
    - Aponta para o ulimo commit
    - Em cada commit, o apontador anda para a frente

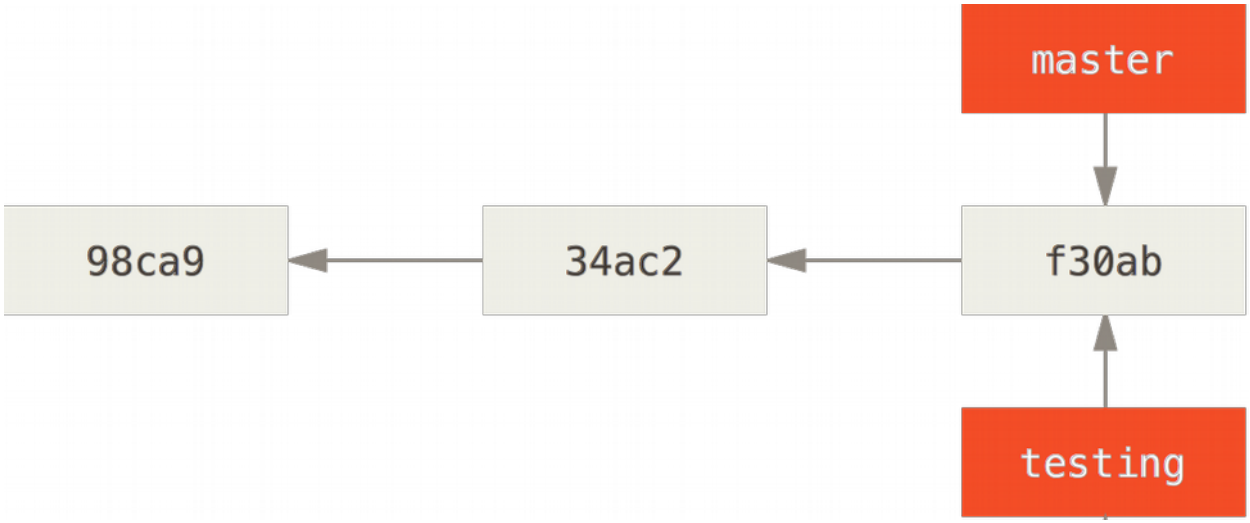
Branches

- Exemplo



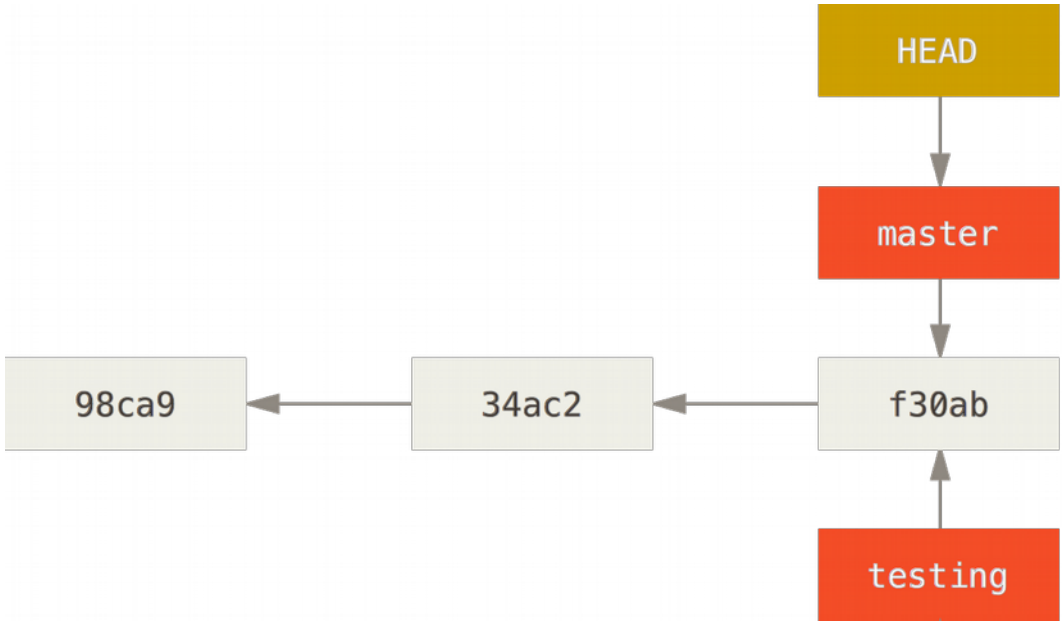
Branches

- Criar um branch novo
  - `git branch testing`



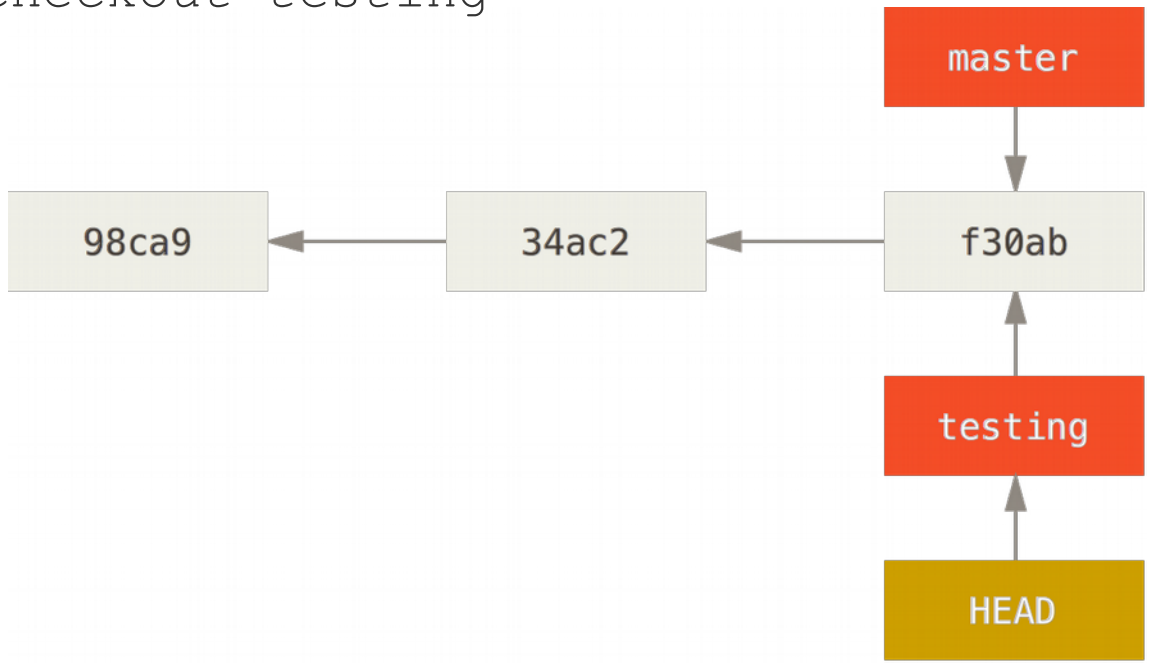
Branches

- Como é que o git sabe qual o branch onde estamos a trabalhar?
  - `HEAD` → Apontador especial para o branch actual



Branches

- Mudar de Branch
  - `git checkout testing`

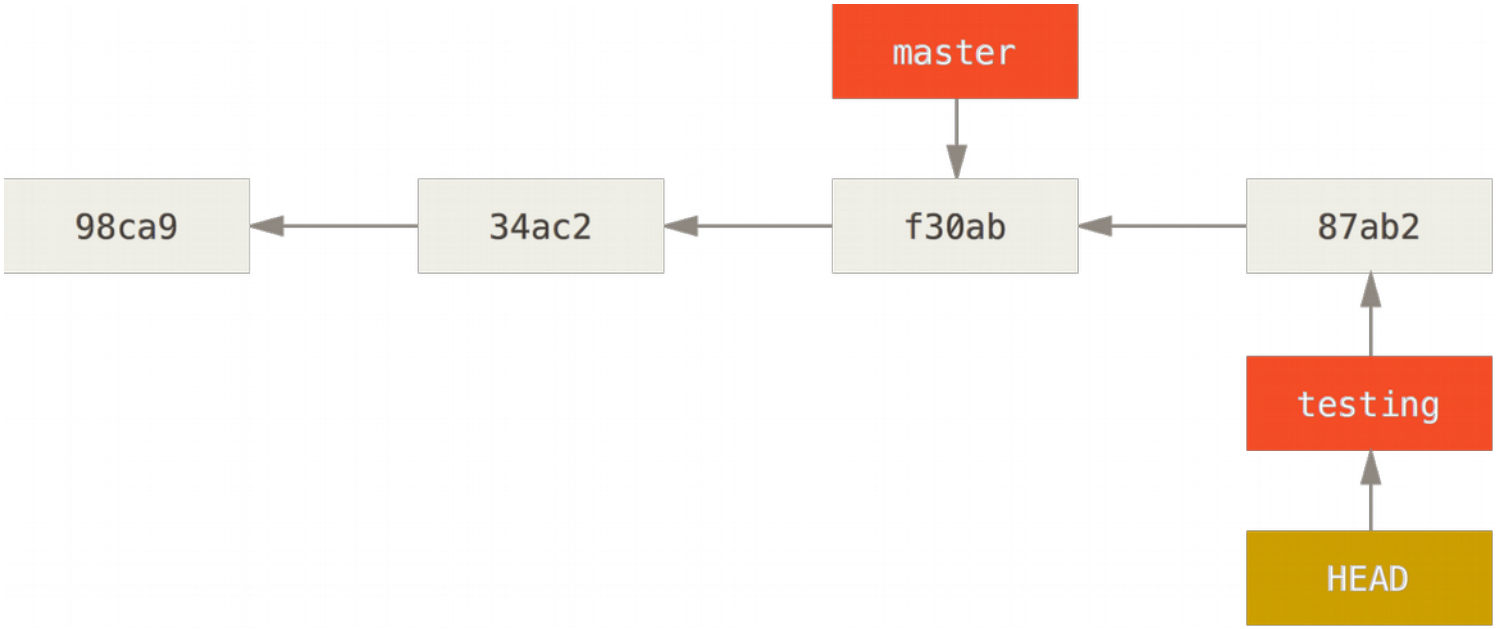


  - O que acontece se fizer um commit novo?

Branches

- Mudar de branch

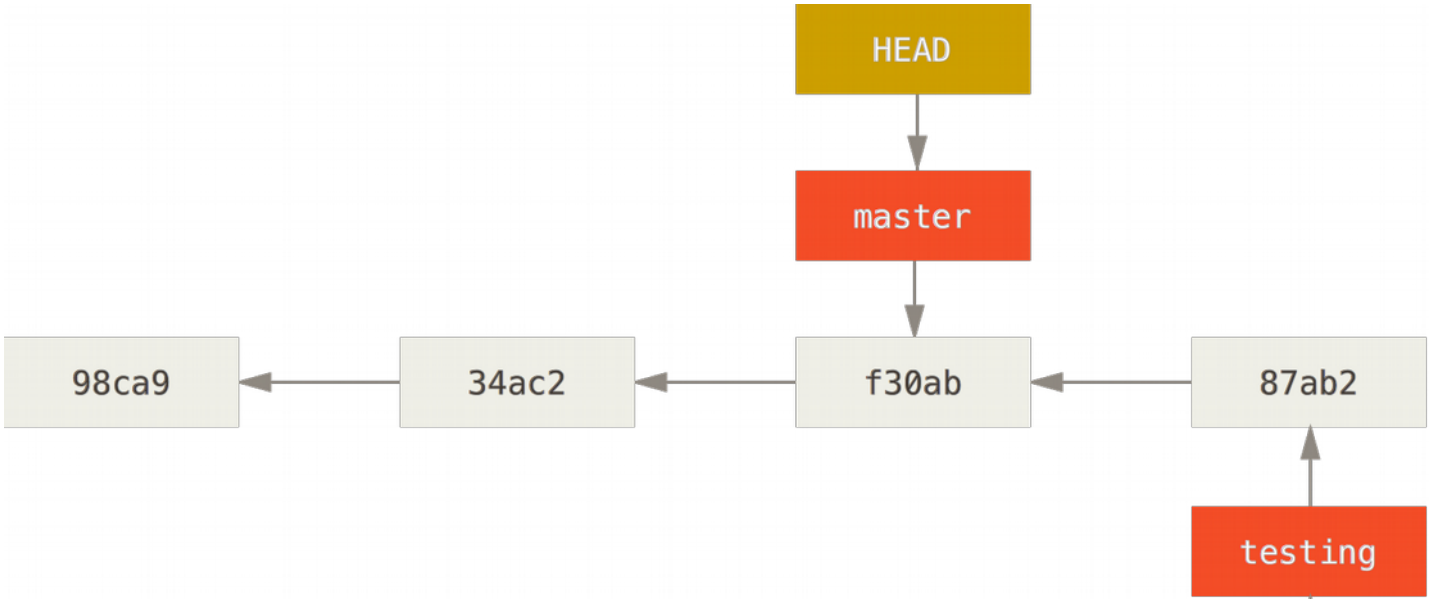
```
$ emacs test.rb
$ git commit -a -m 'made a change'
```



Branches

- Mudar de branch

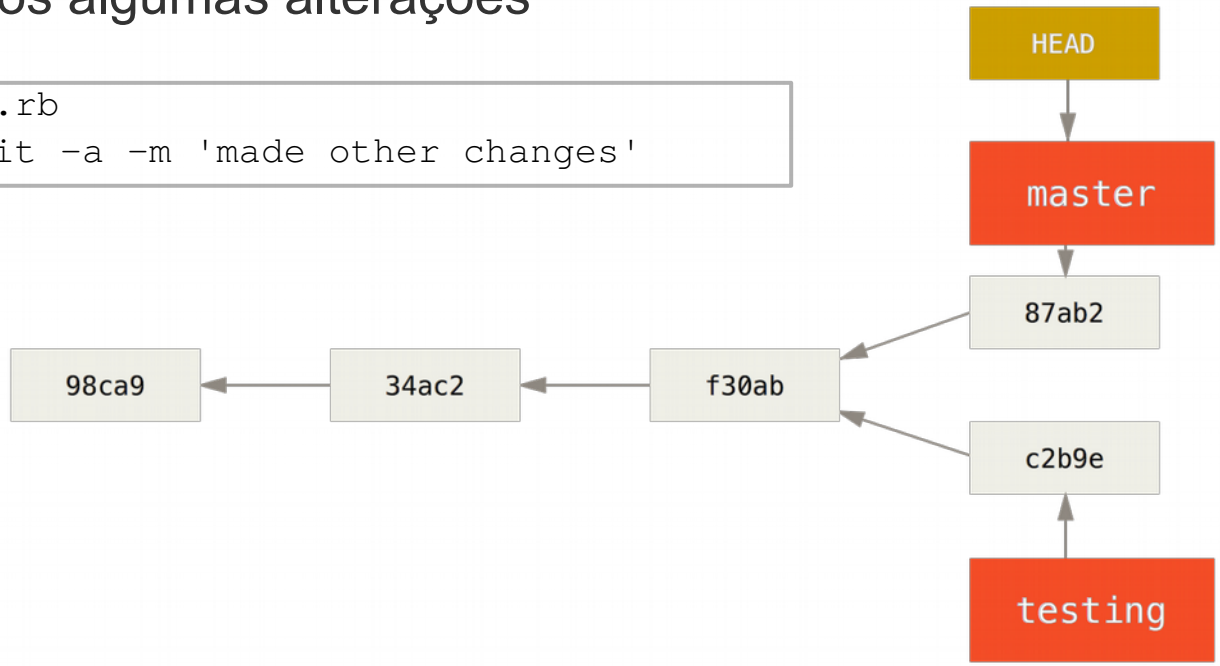
```
$ git checkout master
```



Branches

- Mudar de branch
  - Fazemos algumas alterações

```
$ vim test.rb
$ git commit -a -m 'made other changes'
```

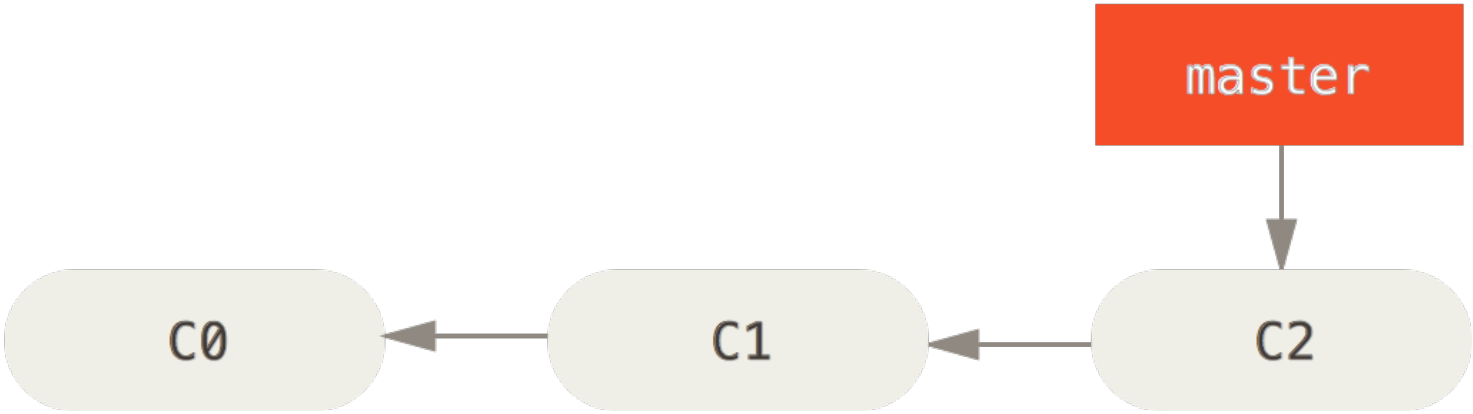


Branching e *Merging*

- *Branching* e *Merging*
  - Cenário
    - Atribuida uma tarefa/issue
    - Cria-se um branch para trabalhar nessa tarefa/issue
    - Trabalha-se nessa tarefa/issue
    - Recebe-se “ordens” para trabalhar num problema crítico!
      - Muda-se para o branch de “produção”
      - Cria-se um branch para o problema critico
      - Trabalha-se no problema
      - Faz-se merge do branch criado para resolver o problema
      - Muda-se para o branch da tarefa/issue original

Branching e Merging

- Ponto de partida



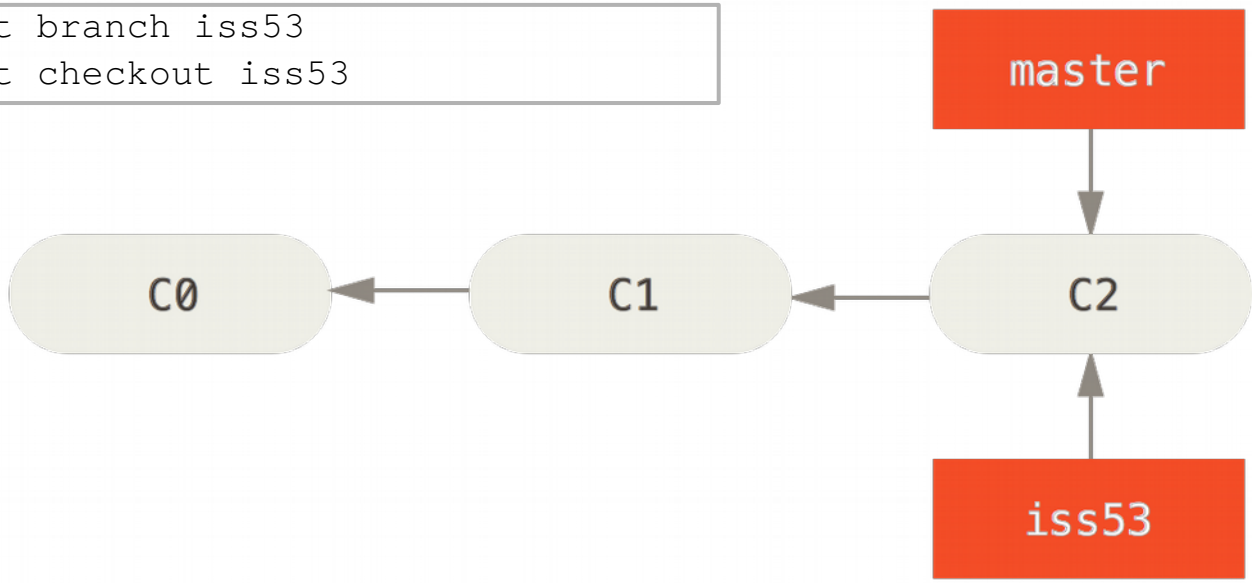
Branching e Merging

- Mudar para um novo branch

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

- **atalho para**

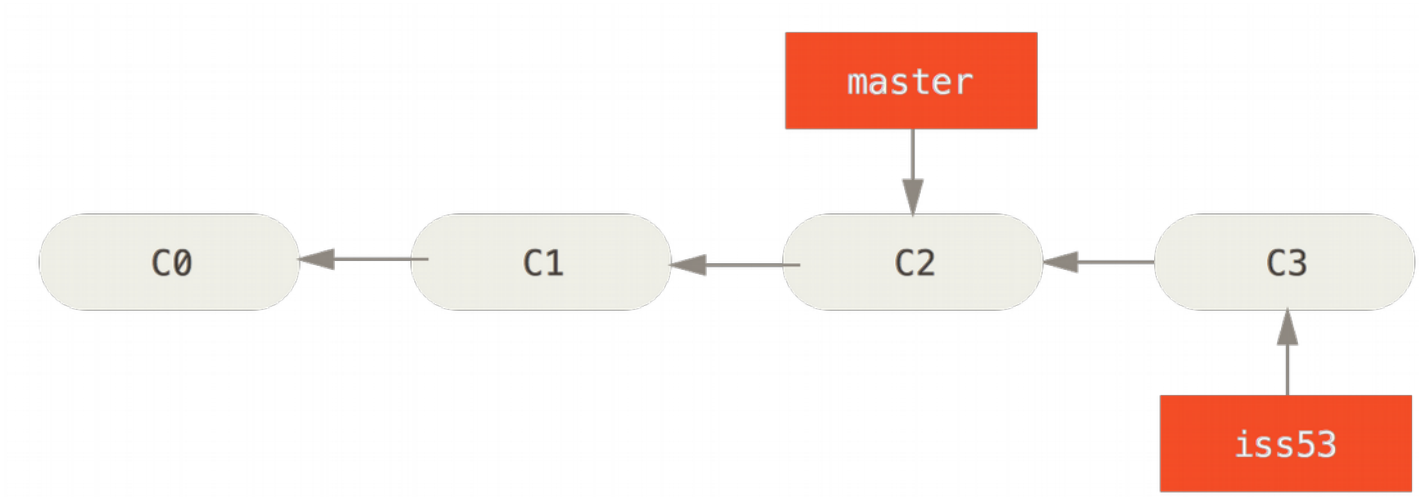
```
$ git branch iss53
$ git checkout iss53
```



Branching e Merging

- Fazer alterações e um commit

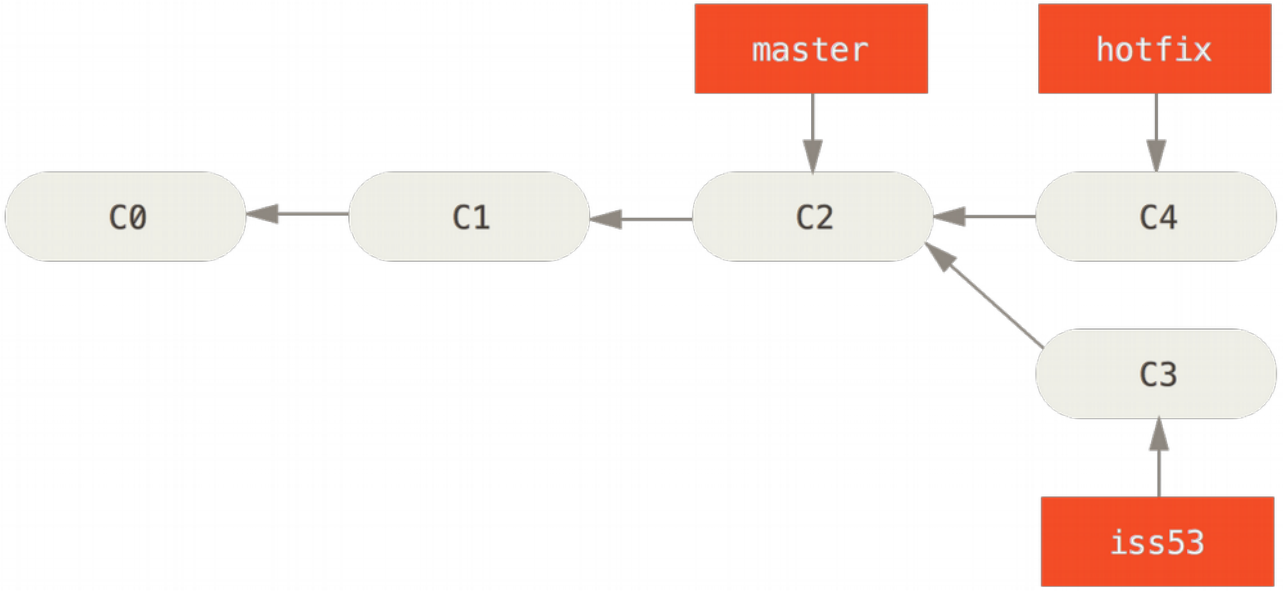
```
$ vim index.html
$ git commit -a -m 'added a new footer [issue 53]'
```



Branching e Merging

- Lidar com o problema “crítico”

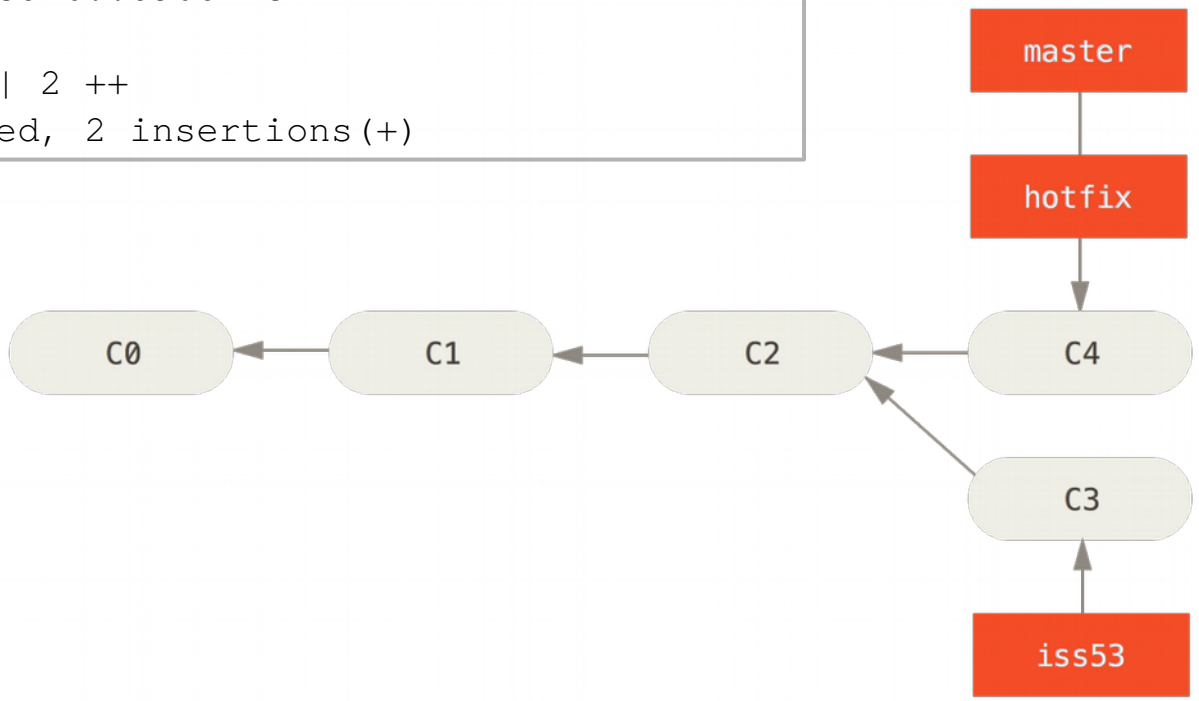
```
$ git checkout -b hotfix
Switched to a new branch 'hotfix'
$ vim index.html
$ git commit -a -m 'fixed the broken email address'
[hotfix 1fb7853] fixed the broken email address
 1 file changed, 2 insertions(+)
```



Branching e Merging

- Fazer merge do “problema crítico” no branch master

```
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast-forward
 index.html | 2 ++
 1 file changed, 2 insertions(+)
```



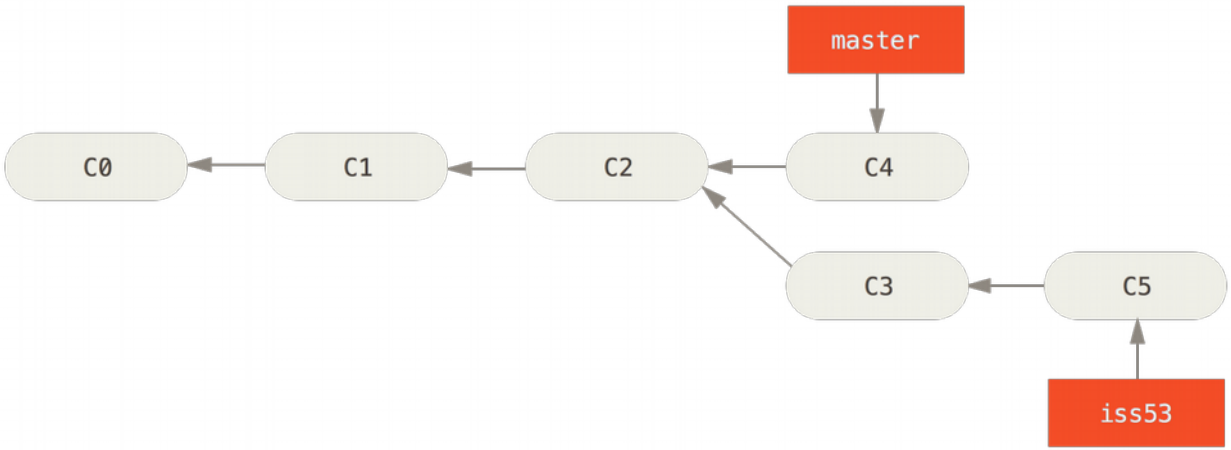
Branching e Merging

- Apagar o branch do “problema crírito”

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

- Mudar para o branch da “tarefa original”

```
$ git checkout iss53
Switched to branch "iss53"
$ vim index.html
$ git commit -a -m 'finished the new footer [issue 53]'
[iss53 ad82d7a] finished the new footer [issue 53]
1 file changed, 1 insertion(+)
```



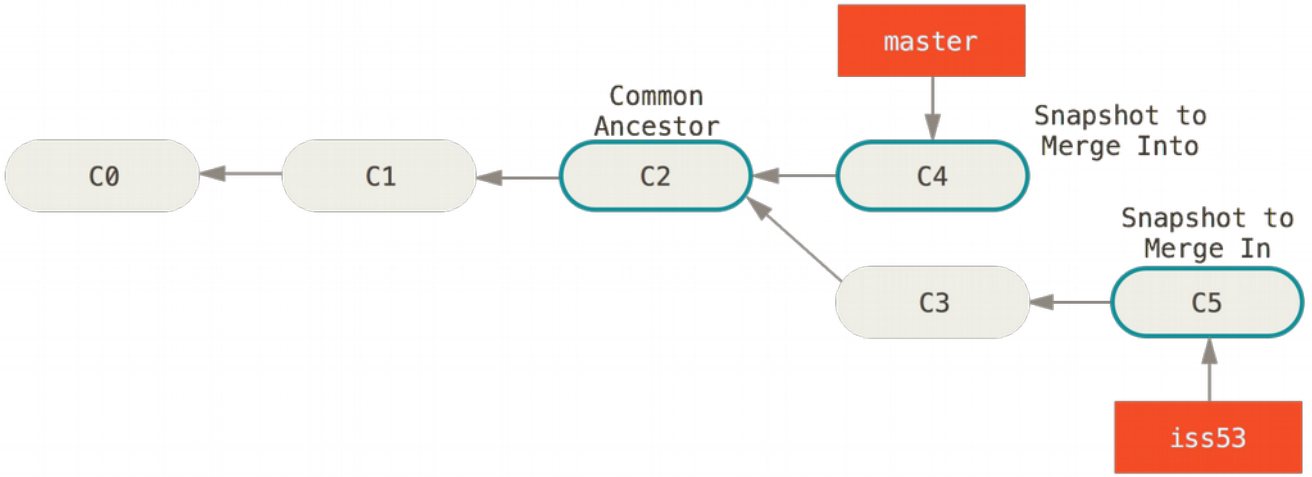
Branching e *Merging*

- Fazer *merge* do branch da tarefa original com o branch master
  - branch actual
    - issue53
  - como fazer o merge
    - merge do branch master para o branch issue53 **OU**
    - **merge do branch issue53 para o branch master**

Branching e Merging

- **merge do branch issue53 no branch master**

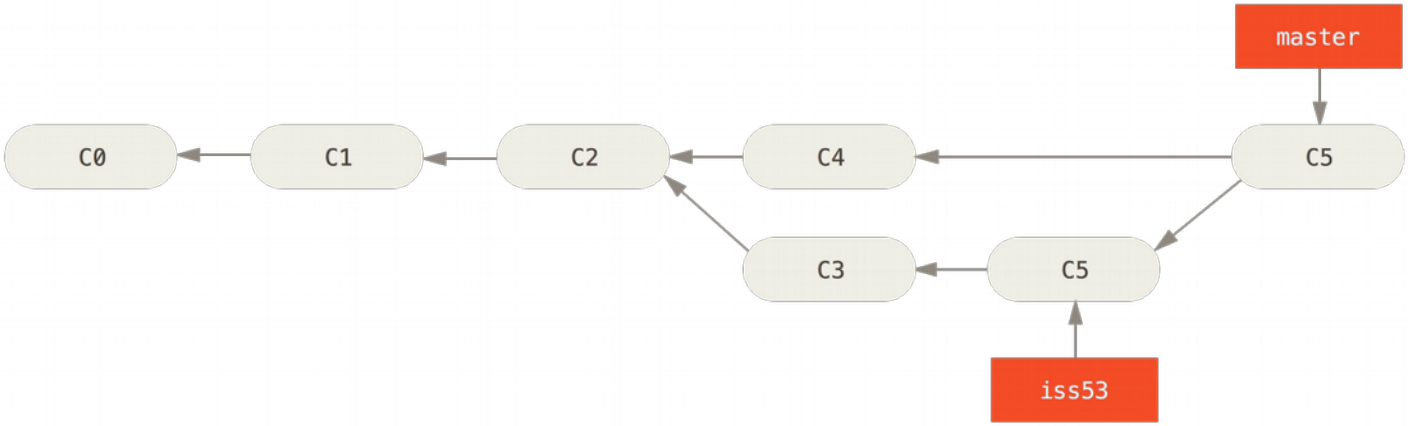
```
$ git checkout master
Switched to branch 'master'
$ git merge iss53
Merge made by the 'recursive' strategy.
index.html |    1 +
1 file changed, 1 insertion(+)
```



Branching e Merging

- **merge do branch issue53 no branch master**

```
$ git checkout master
Switched to branch 'master'
$ git merge iss53
Merge made by the 'recursive' strategy.
index.html |    1 +
1 file changed, 1 insertion(+)
```



Branching e Merging

- Conflitos
  - Quando um merge não corre bem
  - Merge não cria um commit automático

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

```
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file>..." to mark resolution)

    both modified:      index.html

no changes added to commit (use "git add" and/or "git commit -a")
```

Branching e Merging

- Conflitos

```
<<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=======
<div id="footer">;
 please contact us at support@github.com
</div>;
>>>>>>>; iss53:index.html
```

- Above “======” → HEAD (master branch)
- Below “======” → branch iss54

Branching e Merging

- Conflitos
  - Reparar o conflito

```
$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse diffmerge
ecmerge p4merge araxis bc3 codecompare vimdiff emerge
Merging:
index.html

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

Branching e Merging

- Conflitos
  - Após reparar o conflito
    - Alterções são “marcadas” para o próximo commit

```
$ git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

    modified:   index.html
```

Basic Branching and Merging

- Conflitos
  - Fazer commit das alterações: `git commit`

```
Merge branch 'iss53'

Conflicts:
    index.html
#
# It looks like you may be committing a merge.
# If this is not correct, please remove the file
#    .git/MERGE_HEAD
# and try again.


# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# All conflicts fixed but you are still merging.
#
# Changes to be committed:
#    modified:   index.html
```

Gestão de branches

- Listar branches

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

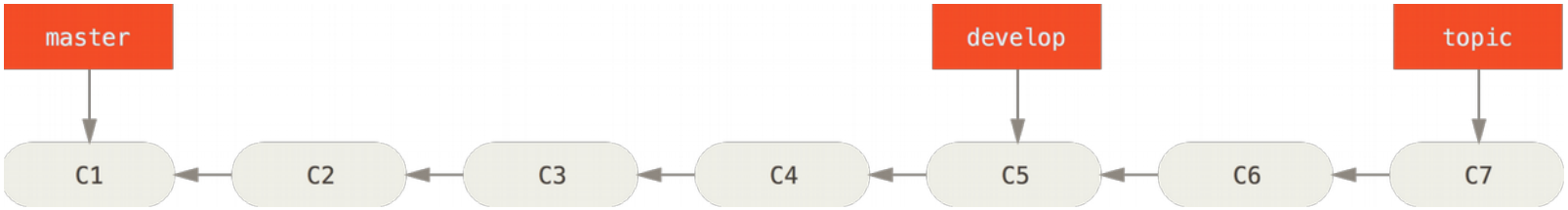
```
$ git branch -v
  iss53   93b412c fix javascript issue
* master  7a98805 Merge branch 'iss53'
  testing 782fd34 add scott to the author list in the readmes
```

Fluxo de branches

- Branches de longa duração
  - Vários branches
    - master
      - branch principal
      - branch de produção
    - develop
      - branch de desenvolvimento
      - código “estável”
      - testar estabilidade e integração
      - usado para fazer merge dos branches dos issues/tarefas
    - topic
      - para desenvolver issues/tarefas

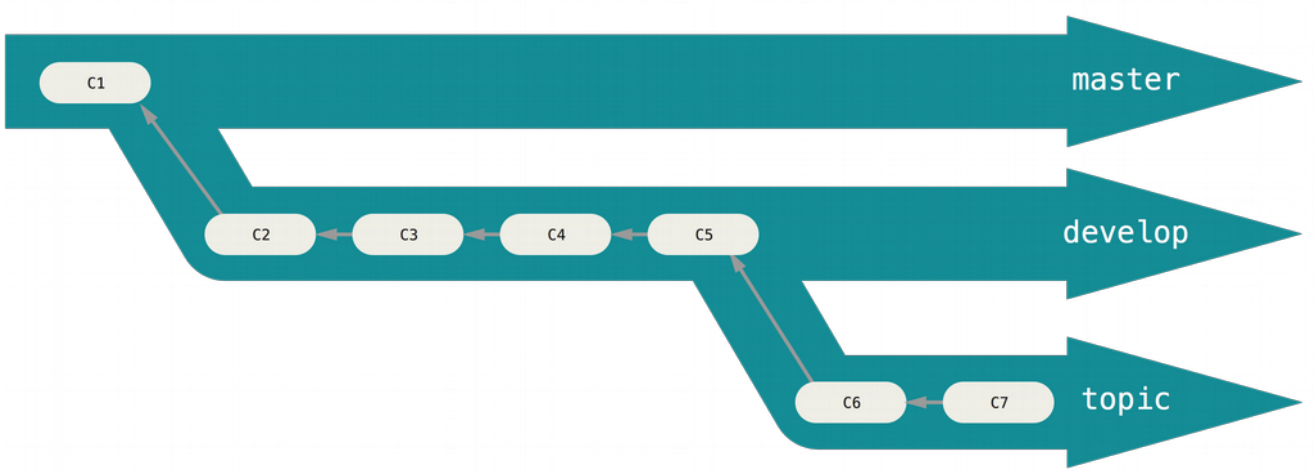
Fluxo de branches

- Branches de longa duração



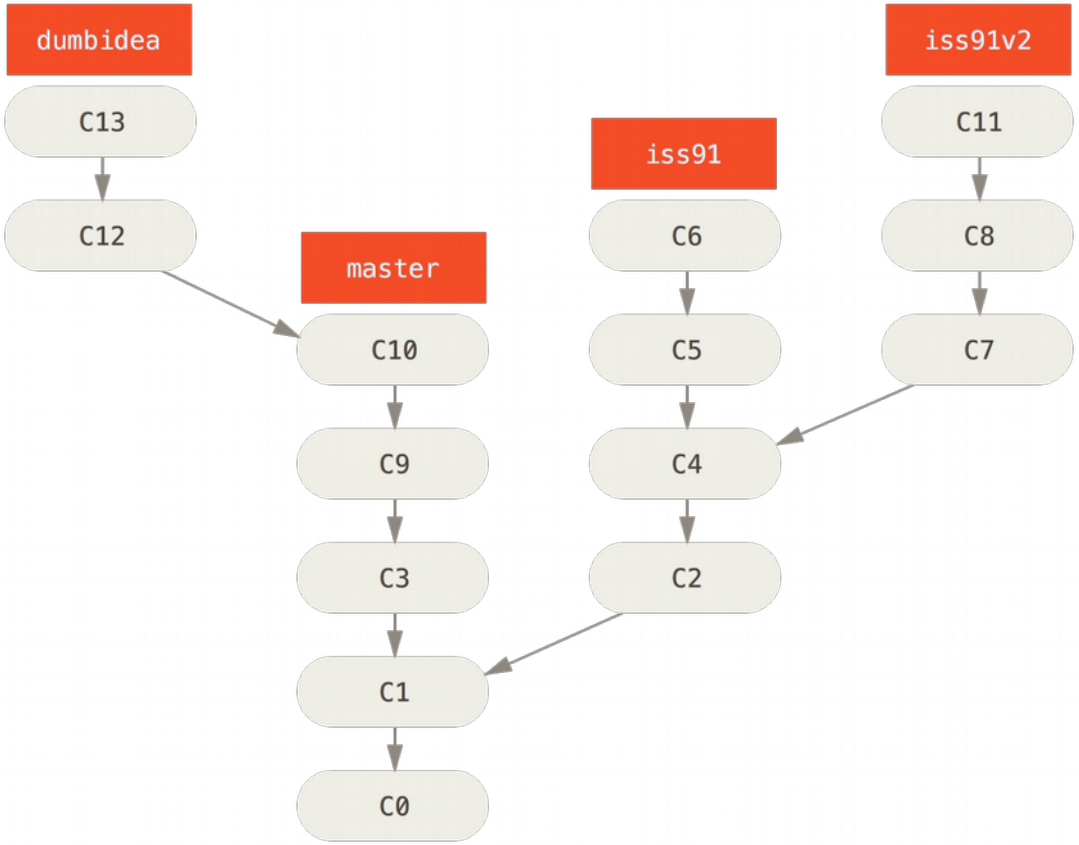
Fluxo de branches

- Branches de longa duração



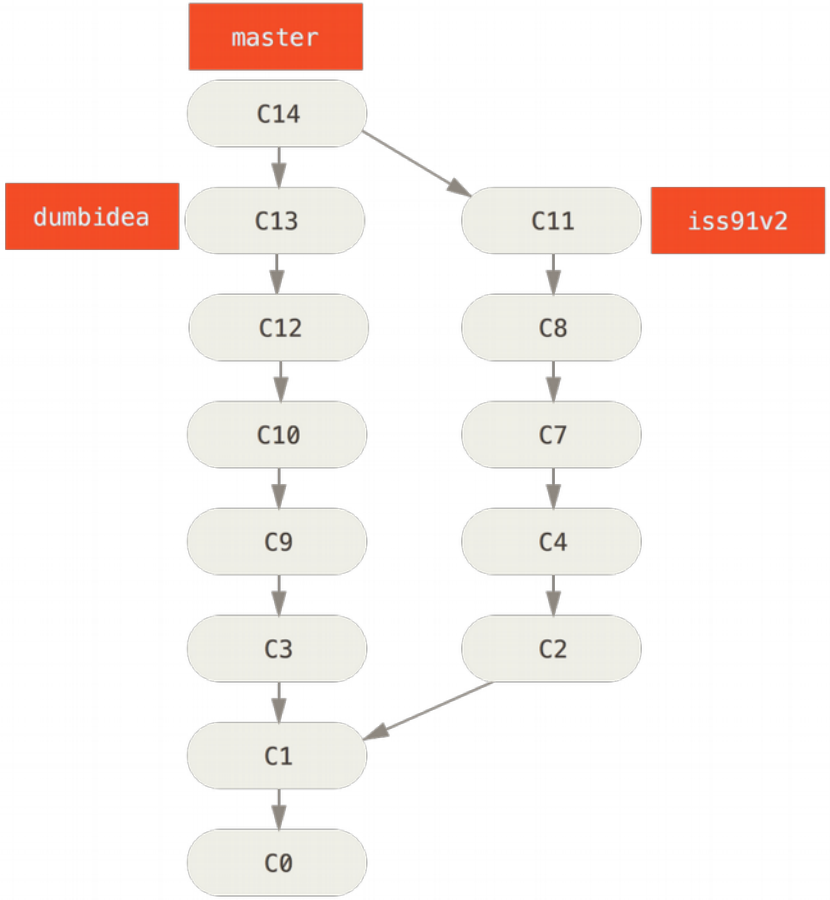
Fluxo de branches

- Topic branches



Fluxo de branches

- Gostámos do “iss91v2” e do “dumbideia”
  - merge do “dumbidea” para o master
  - apagamos o “iss91”
    - perdemos o “C5” e o “C6”
  - merge do “iss91v2” para o master



Bibliography

- Chacon, Scott, and Ben Straub. *Pro git*. Apress, 2014.
  - https://git-scm.com/book/en/v2