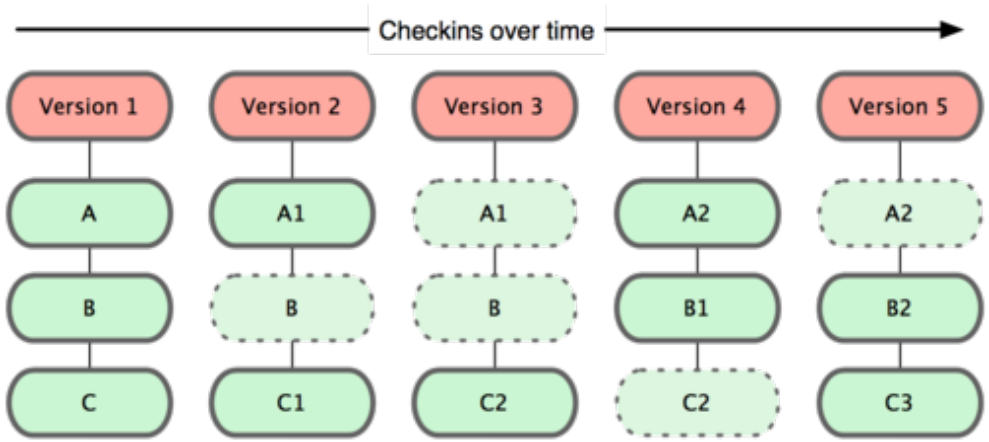
SCM

- historiser les modifications des fichiers d’un projet
- gérer plusieurs versions
- collaborer à plusieurs intervenants
- Source Code Managememnt
- Software Configuration Management

GIT

- SCM du noyau linux
- Distribué
- Gestion des changements sur une arborescence de fichiers
- une version : snapshot d’un projet (pas différentiel par rapport à la version précédente)
- gestion historique et branche locale : RAPIDE !



Git : TP

configurer : mail et nom

`--global` : `~/.gitconfig`

sinon : `.git/config` du projet

```
$ git config --global user.name "Pierre Gambarotto"
$ git config --global user.email pierre.gambarotto@enseeiht.fr
$ git config --global core.editor vim
$ git config --global merge.tool vimdiff
$ git config --list
user.name=Pierre Gambarotto
user.email=pierre.gambarotto@enseeiht.fr
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
```

Initialiser un depot

```
$ cd /path/to/project

$ git init
Initialized empty Git repository
  in /tmp/project/.git/

$ ls -a
.git
```

Cycle de travail de base

```
$ vim hello.rb # add stuff

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

préparer un commit

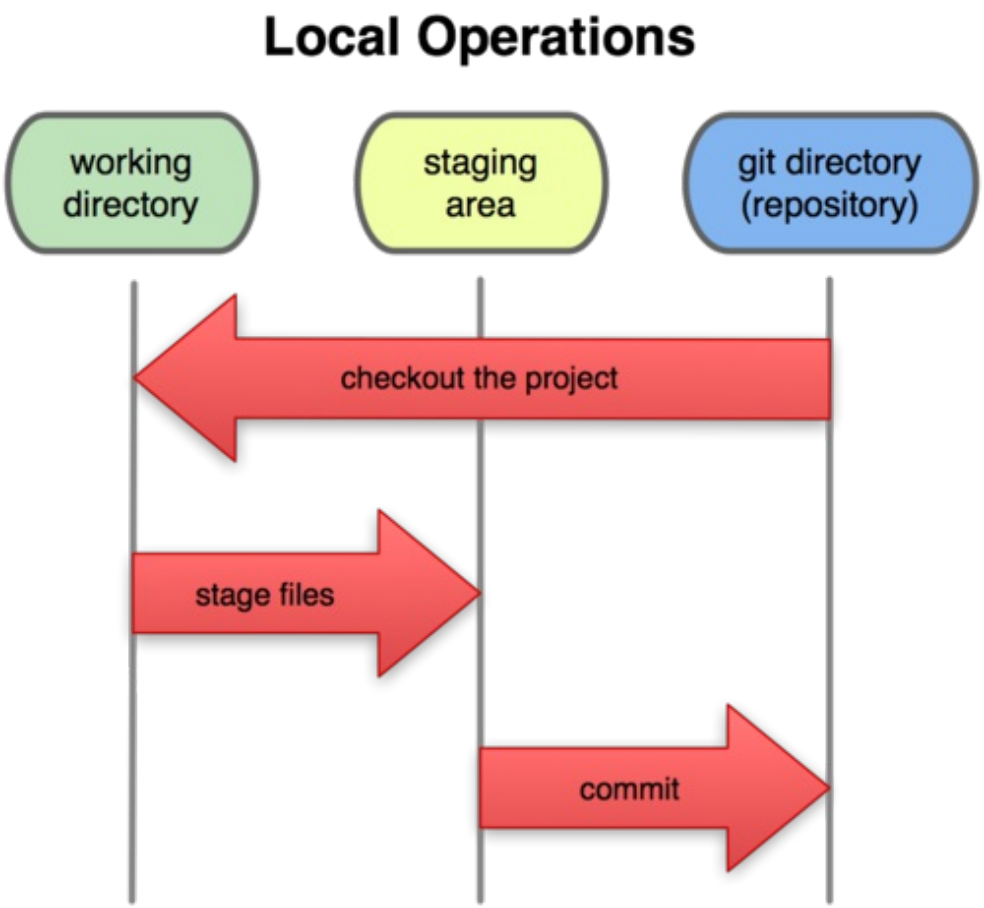
sélectionner les changements

les ajouter au staging

```
$ git add hello.rb
$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
# new file:   hello.rb
#
```

commit

```
$ git commit -m "added first file, wooot"
[master (root-commit) 15aa711] added toto, woooot
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 hello.rb
$ git status
# On branch master
nothing to commit (working directory clean)
```


Local Operations
working directory
staging area
git directory (repository)
checkout the project
stage files
commit

Git à la petite semaine

créer un dépôt : `cd project; git init`

copier un dépot existant :

- url git :
  - local : file:///path/to/projet.git
  - ssh : toto@server:/path/to/projet.git (securisé et authentifié)
  - git : git://server/path/to/projet.git (anonyme)
  - http : équivalent ssh, plus lent

```
git clone toto@server:projet.git

cd projet

ls -a

.git ...
```

travailler : `vim/emacs file ...`

préparer un commit :

`git status` # donne les changements qui ont été fait

`git add ...` # choisir les changements que l'on va prendre en compte dans le commit

`git commit -m message`

reverser ses modifs sur un autre dépot :

`git push`

Git : modèle objet très simple

- objets stockés dans .git/objects
- blob, tree, commit, tag
- tous les objets sont identifiés par des sha1 : signature numérique du contenu sur 38 octets

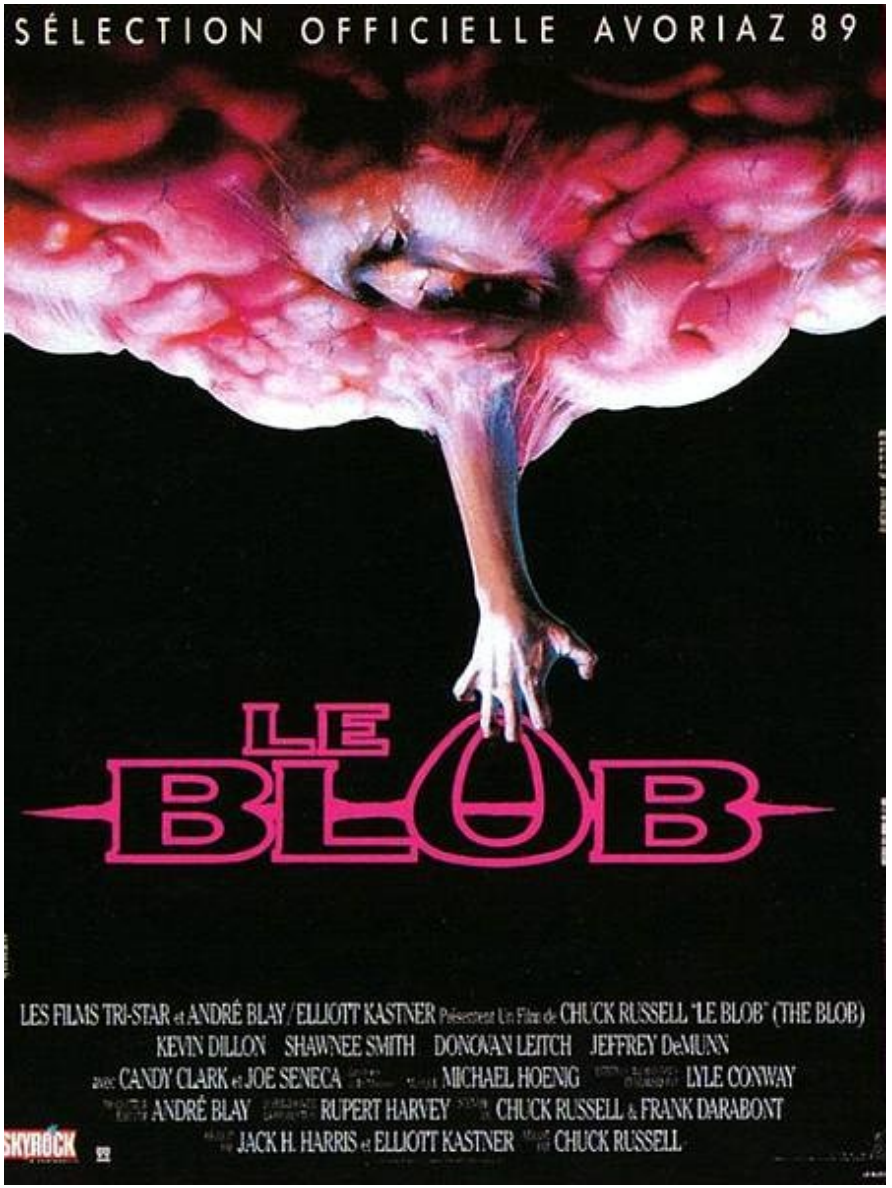
le sha1 est roi

106a93177346290bb12f5ca4e666eb875d321c

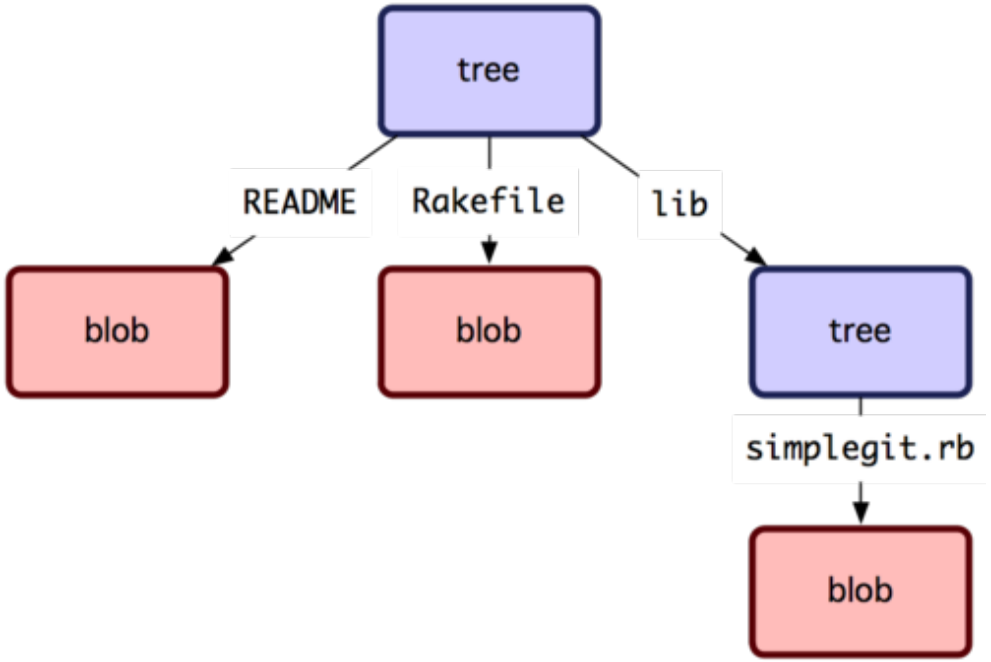
Blob

nom du blob : sha1 du contenu d’un fichier

contenu d’un blob : contenu d’un fichier



Tree



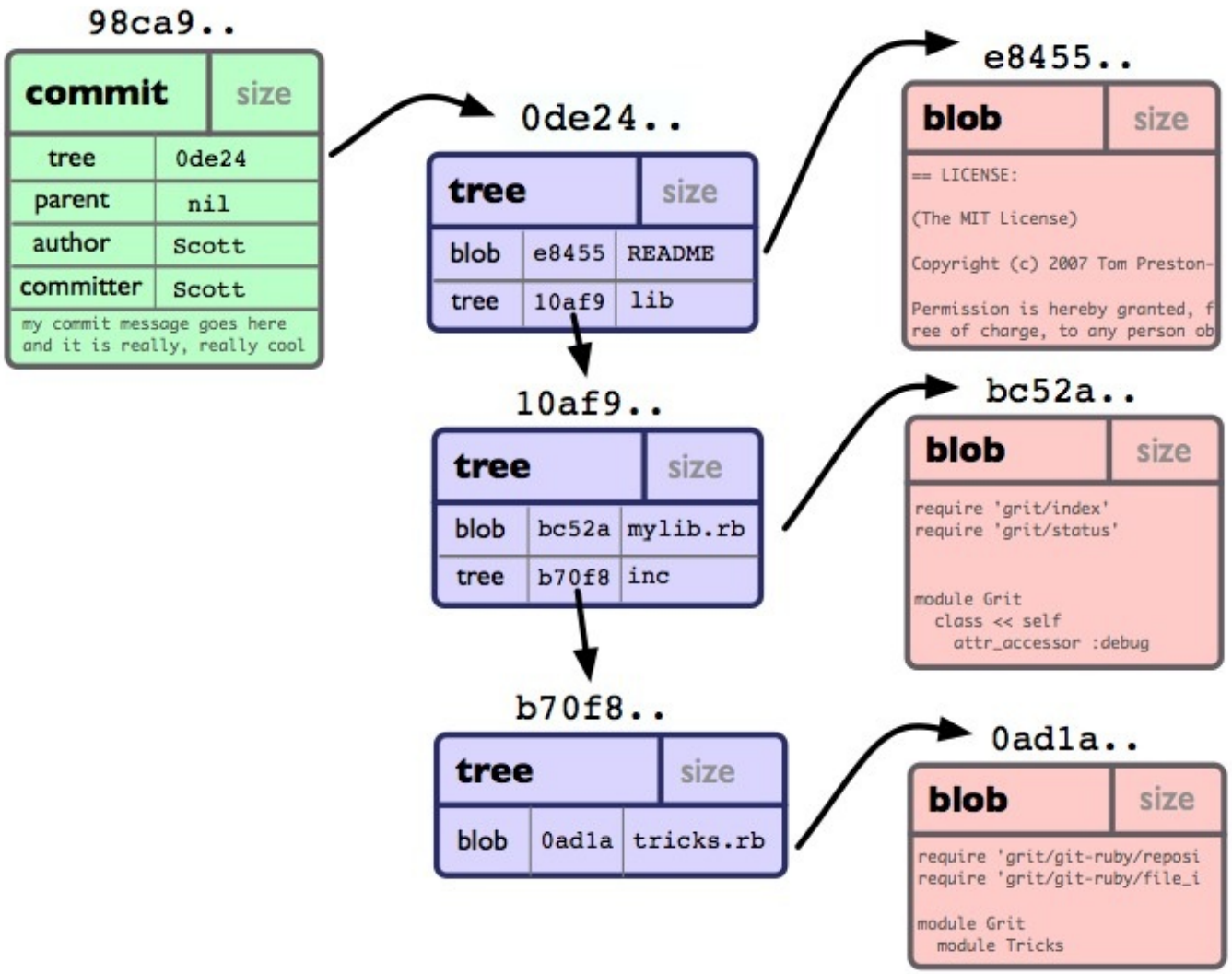
tree : liste de (“blob”, sha1, nom_fichier) ou (tree,

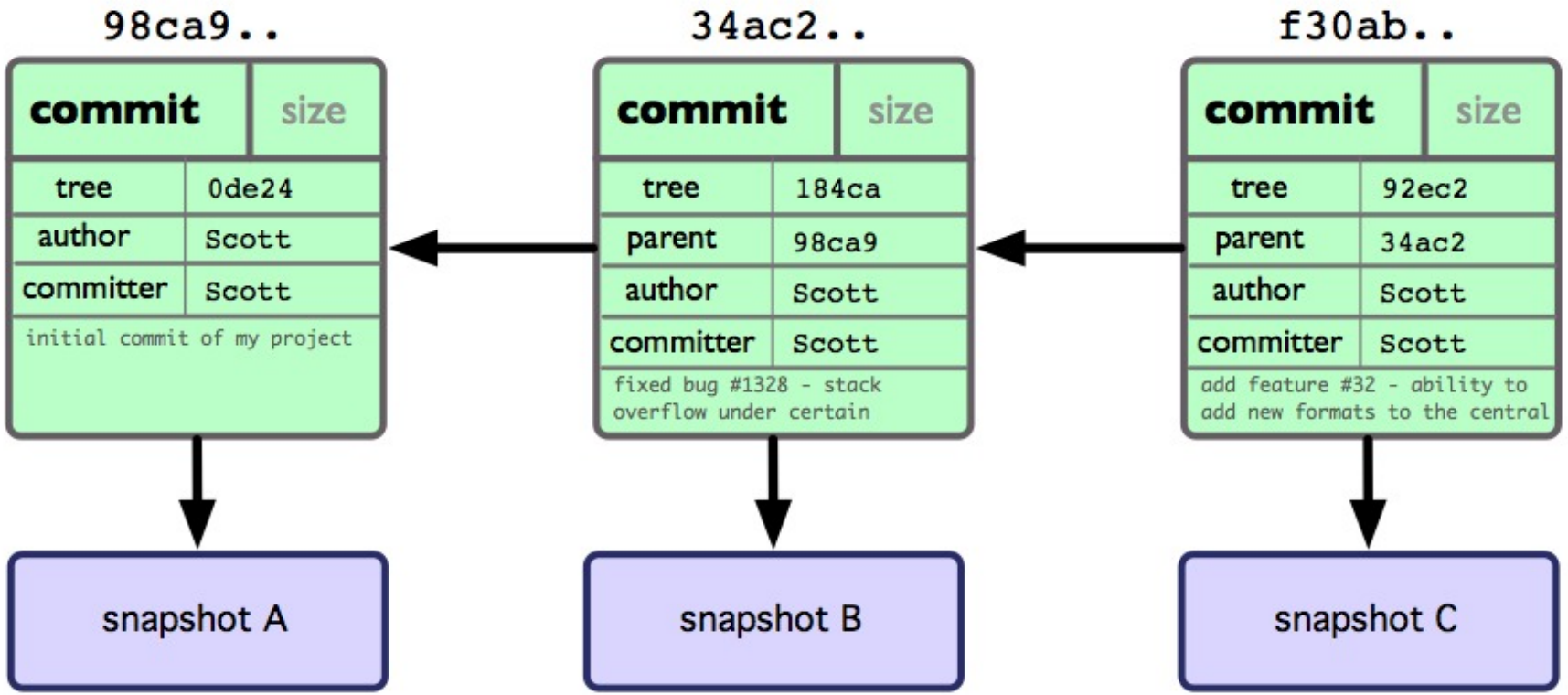
sha1, nom_repertoire)

nom d’un arbre : sha1 du contenu

commit

- tree sha1
- 0+ commits parents (0 : commit initial, 1 : commit classique, merge : 2+ parents)
- message
- nom auteur


98ca9..
commit
size
tree
0de24
parent
nil
author
Scott
committer
Scott
my commit message goes here
and it is really, really cool
0de24..
tree
size
blob
e8455
README
tree
10af9
lib
e8455..
blob
size
== LICENSE:
(The MIT License)
Copyright (c) 2007 Tom Preston-
Permission is hereby granted, f
ree of charge, to any person ob
10af9..
tree
size
blob
bc52a
mylib.rb
tree
b70f8
inc
bc52a..
blob
size
require 'grit/index'
require 'grit/status'
module Grit
class << self
attr_accessor :debug
b70f8..
tree
size
blob
0ad1a
tricks.rb
0ad1a..
blob
size
require 'grit/git-ruby/reposi
require 'grit/git-ruby/file_i
module Grit
module Tricks


98ca9..
commit
size
tree
0de24
author
Scott
committer
Scott
initial commit of my project
34ac2..
commit
size
tree
184ca
parent
98ca9
author
Scott
committer
Scott
fixed bug #1328 - stack overflow under certain
f30ab..
commit
size
tree
92ec2
parent
34ac2
author
Scott
committer
Scott
add feature #32 - ability to add new formats to the central
snapshot A
snapshot B
snapshot C

en mode collaboratif

cloner un dépot existant

```
$ cd /tmp/
$ git clone login@psylocke:/tmp/sample.git
Initialized empty Git repository in /tmp/sample/.git/
remote: Counting objects: 387, done.
remote: Compressing objects: 100% (312/312), done.
remote: Total 387 (delta 141), reused 114 (delta 21)
Receiving objects: 100% (387/387), 149.31 KiB, done.
Resolving deltas: 100% (141/141), done.
```

on suit automatiquement la branche distante

```
$ cat .git/config
[core]
  repositoryformatversion = 0
  filemode = true
  bare = false
  logallrefupdates = true
[remote "origin"]
  url = gamba@psylocke:/tmp/sample.git
  fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
  remote = origin
  merge = refs/heads/master
```

push : commit(local) => depot distant

```
$ git push origin master
Counting objects: 7, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 377 bytes, done.
Total 4 (delta 3), reused 0 (delta 0)
To gamba@psylocke:/tmp/sample.git
   df4959c..115c344  master -> master
```

SCM: Branches

- pour tester sans détruire
- pour faire bosser un stagiaire
- dualité production/développement
- technique : 1 fonctionnalité = 1 branche

Git : Branches

- branche = référence sur un commit : `.git/refs/*`
- branches locale : `.git/refs/heads/*`
- branche remote : `.git/refs/remotes/`

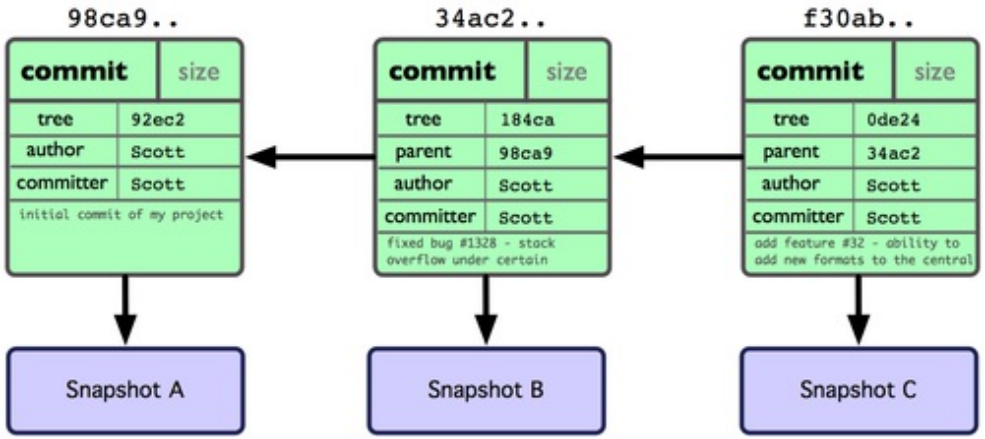
Attention : les branches remote sont des branches locales, correspondant à une copie d’une branche d’un autre dépot.

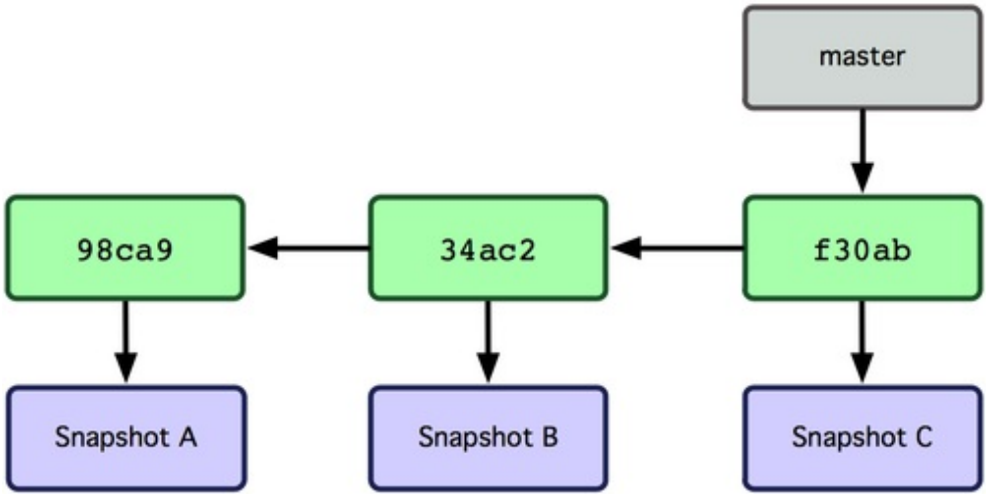
branches spéciales

- `master` : le nom de la branche par défaut
- `HEAD` : la branche du répertoire courrant
- `origin/master` : nom par défaut d'une branche distante
- branche distante : (remote)/(branche) : copie locale d'une branche d'un dépôt distant

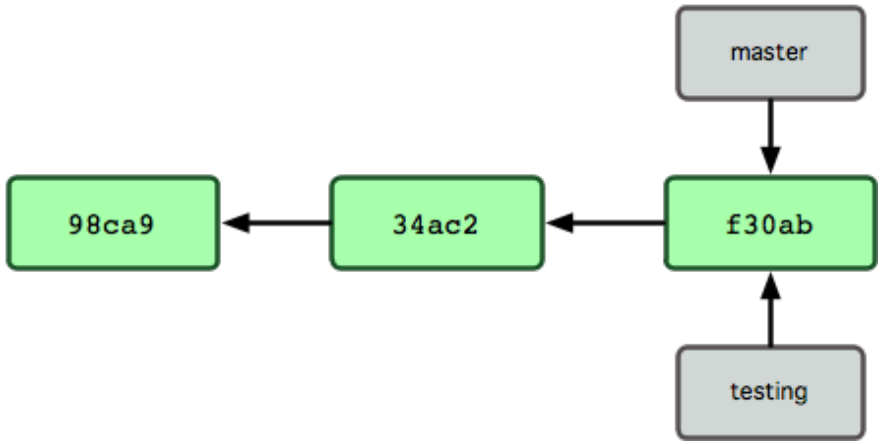
Git : Branches et commande

- `git branch` : liste les branches
- `git branch testing` : crée la branche `testing`, identique à la branche courante
- `git checkout testing` : modifie le répertoire courant pour refléter l'état d'une branche
- -b : crée la branche à partir de l'actuelle et checkout

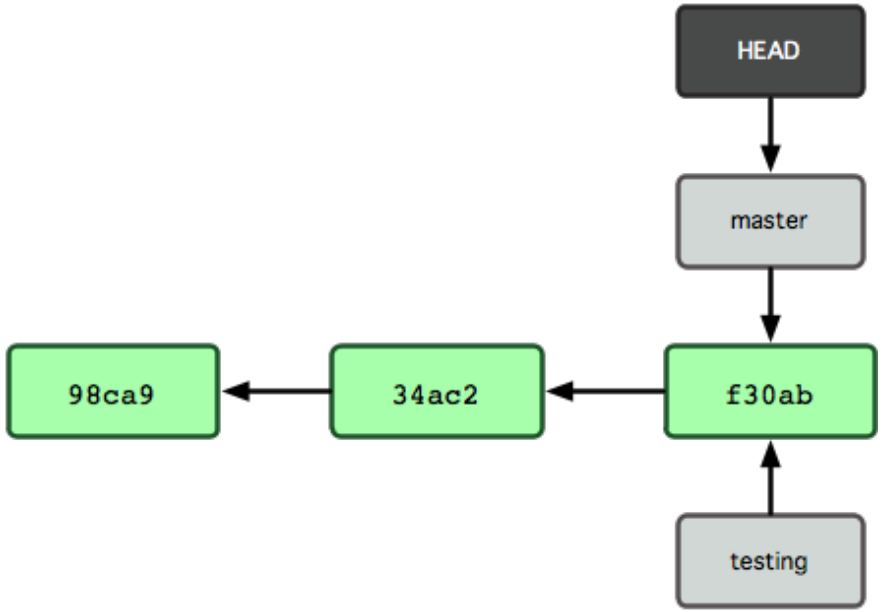

98ca9..
commit
size
tree
92ec2
author
Scott
committer
Scott
initial commit of my project
34ac2..
commit
size
tree
184ca
parent
98ca9
author
Scott
committer
Scott
fixed bug #1328 - stack overflow under certain
f30ab..
commit
size
tree
0de24
parent
34ac2
author
Scott
committer
Scott
add feature #32 - ability to add new formats to the central
Snapshot A
Snapshot B
Snapshot C


master
98ca9
34ac2
f30ab
Snapshot A
Snapshot B
Snapshot C

$ git branch testing



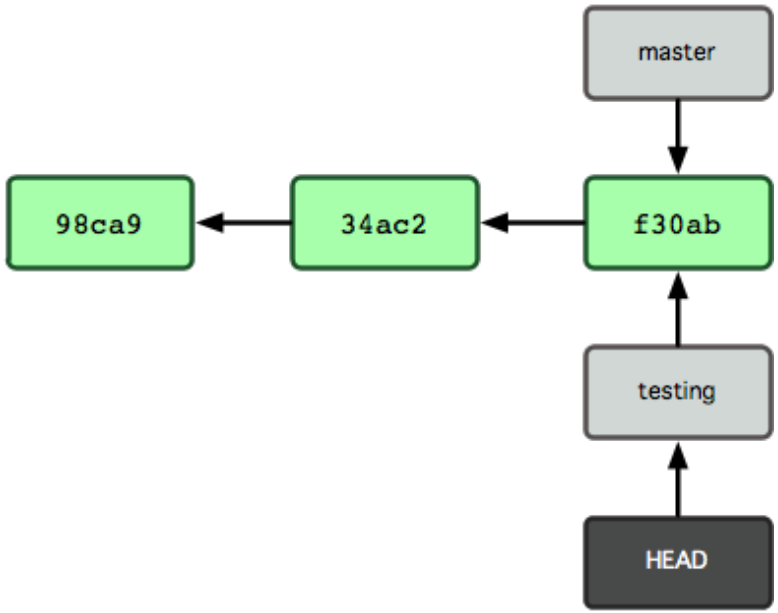
branche du répertoire courant : HEAD



changer HEAD

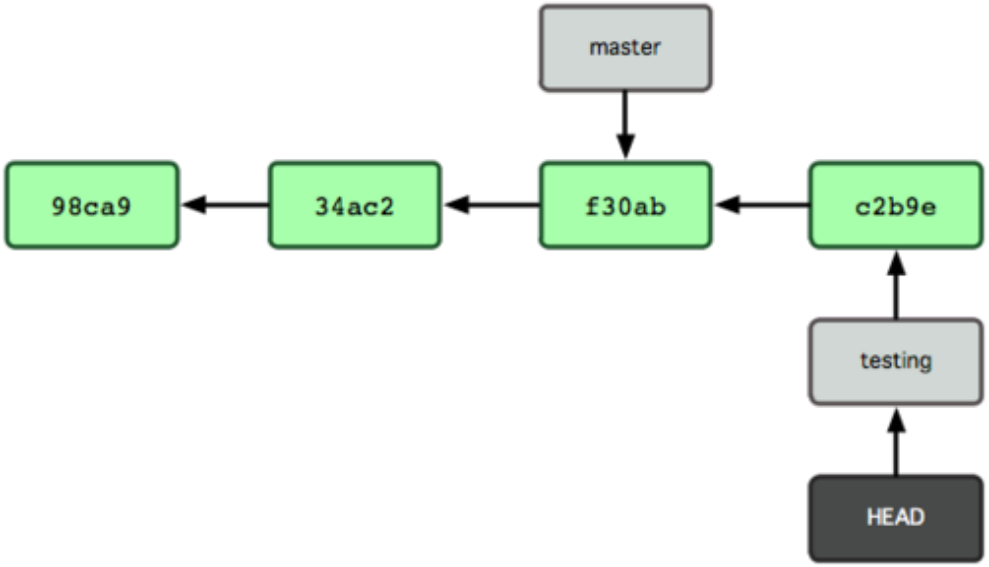
changer le répertoire courant

```
$ git checkout testing
```



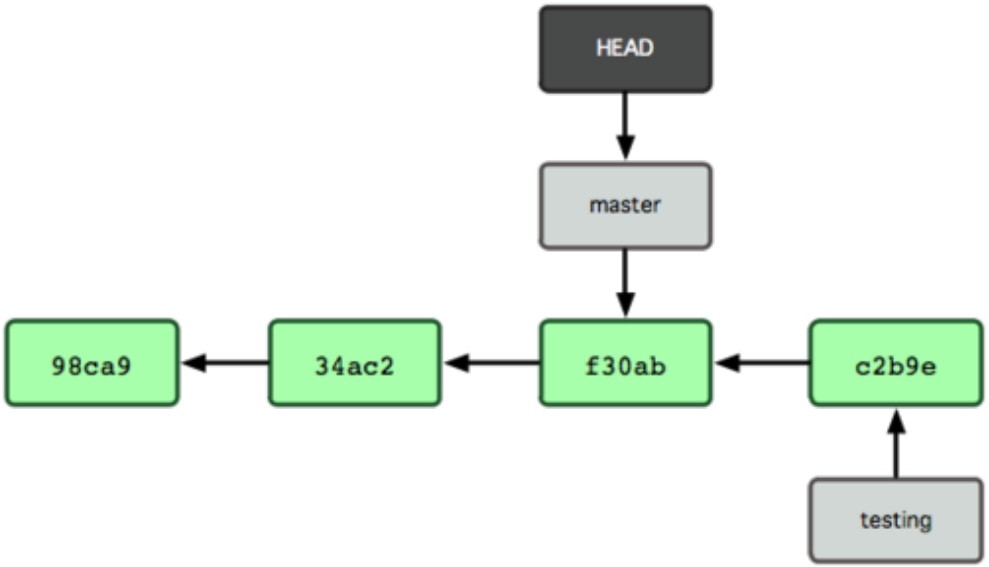
```
$ vim test.rb
```

```
$ git commit -a -m 'made a change'
```



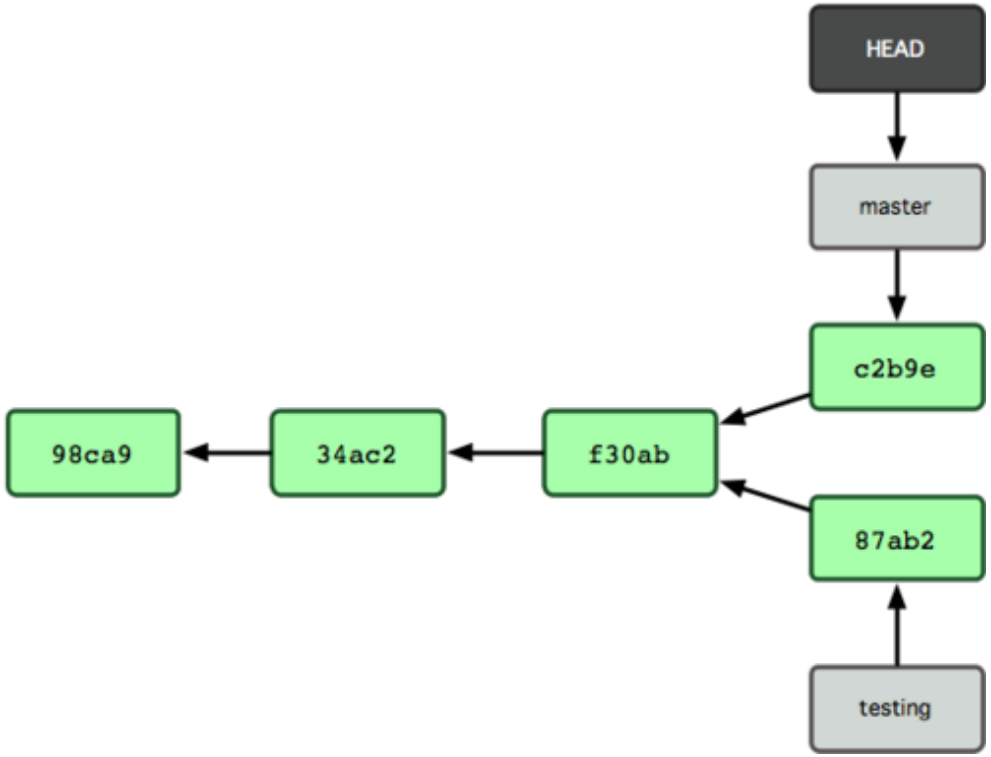
retour sur la branche principale

```
$ git checkout master
```



```
$ vim test.rb
```

```
$ git commit -a -m 'made another change'
```

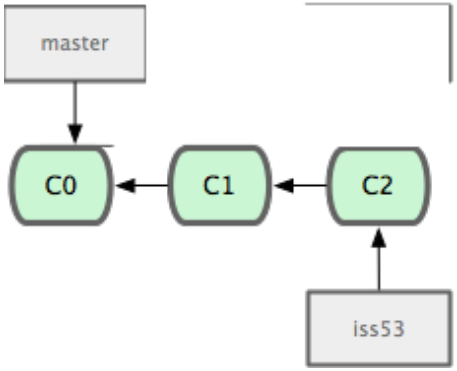


MERGE

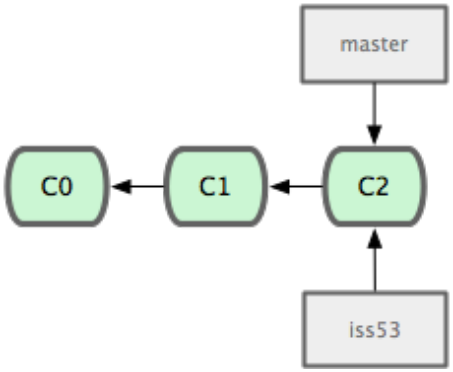
- fusionner les changements d’une branche dans une autre

- `$ git checkout branche_resultat`
- `$ git merge other_branch`

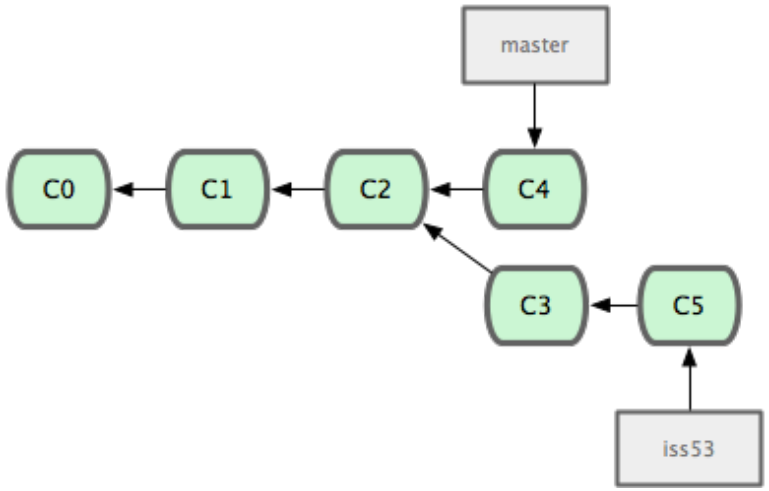
fast forward merge



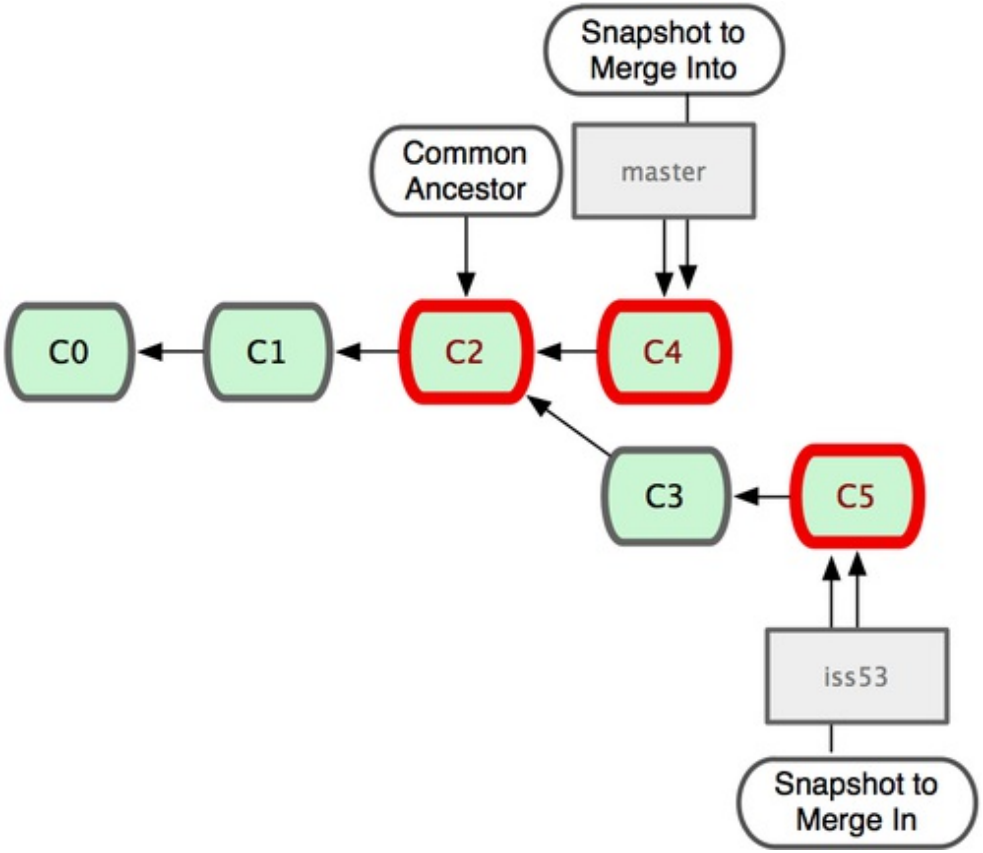

```
$ git checkout master

$ git merge iss53
```



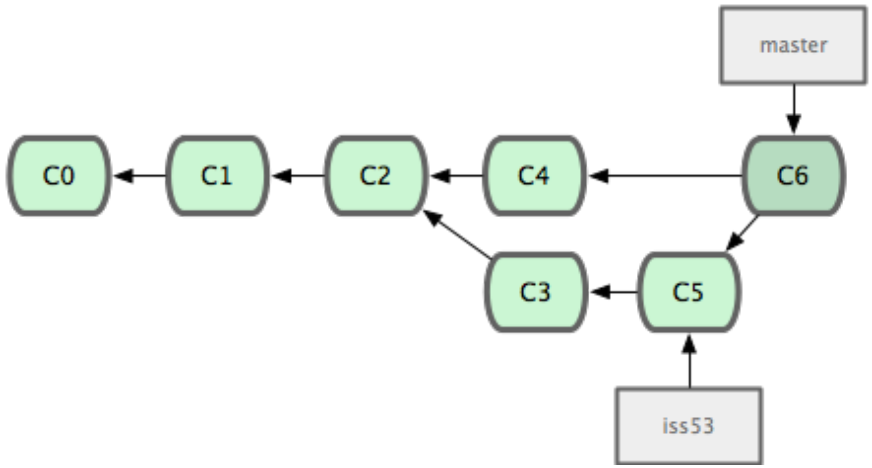
classic merge



classic merge



classic merge



Conflit

```
$ git merge iss534
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then
commit the result.

# créé .git/MERGE_HEAD
```

où ?

```
$ git status
index.html: needs merge
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will
# be committed)
#   (use "git checkout -- <file>..." to discard
# changes in working directory)
#
# unmerged:   index.html
#
```

Résolution :

- éditer le fichier
- repères pour trouver le problème : `<<<<<<< ... ======== .... >>>>>>>>`

résolution

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

finir le merge

```
$ git commit
# message du commit par défaut :
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.
```

Pour aller plus loin

- Git Immersion
- Pro Git
- Github pour tester
- à utiliser pour toutes vos éditions

Les schémas des blob, commit, tree sont extraits du livre Pro Git.

Exercice en solo

- Créer un compte gratuit sur Github

1. Créez un projet `tuto_git` sur votre ordinateur, passez-le sous git
2. Créez un fichier README avec du texte.
3. Créez un dépôt nommé `tuto_git` sur votre compte github.
4. Associez votre dépôt local à votre dépôt distant
5. `git push origin master` !
6. Reprenez toutes les commandes que vous venez de taper; pour chaque ligne, faites une modification dans le fichier README, et faites un commit avec une description adéquate.
7. Refaites un push.
8. Envoyez un mail à `pierre.gambarotto@enseeiht.fr` comportant un lien vers le dépôt sur github.

Pour vous aider à démarrer, consultez la documentation de github, en particulier :

- Connecter un dépôt local à son compte github
- Créer un dépôt sur github

Last but not least : parcourez Git Immersion

Pour le cours suivant

- installer `rspec` :

  gem install rspec