# UN CUENTO DE

# TRES ÁRBOLES

## una tarde mágica con Scott Chacón

# git.io/tres-arboles

# introducción

# introducción

# reset

# reset

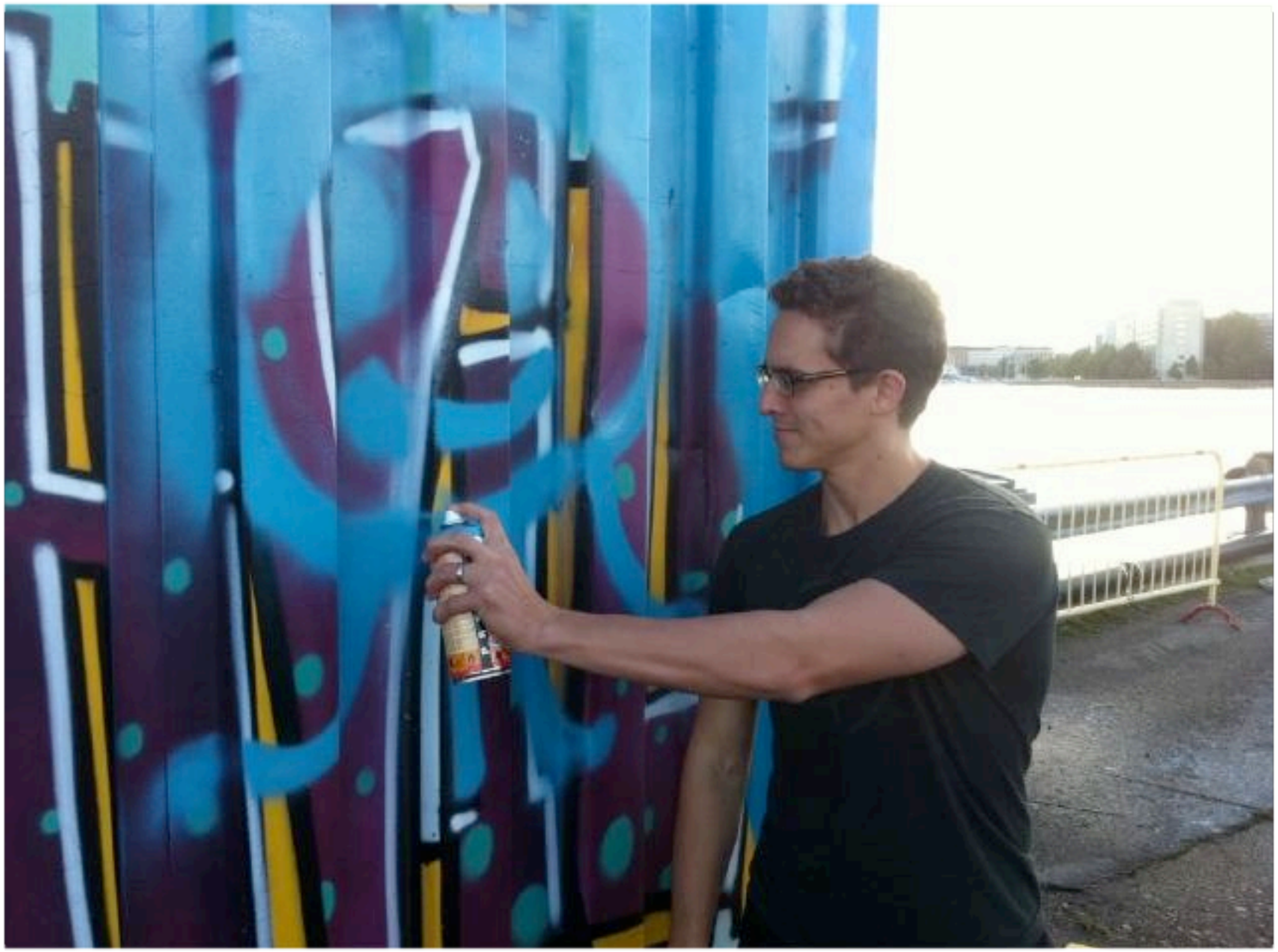
# usan reset?

# UN CUENTO DE

# TRES ÁRBOLES

## una tarde mágica con Scott Chacón

mi



# scott chacon


github
SOCIAL CODING
CIO


Home | About Git | Documentation | Download | Tools & Hosting

## Git is...

Git is a **free & open source, distributed version control system** designed to handle everything from small to very large projects with speed and efficiency.

**Every Git clone is a full-fledged repository** with complete history and full revision tracking capabilities, not dependent on network access or a central server. **Branching and merging are fast** and easy to do.

Git is used for version control of files, much like tools such as Mercurial, Bazaar, Subversion, CVS, Perforce, and Team Foundation Server.

## Projects using Git

- Git
- Linux Kernel
- Perl
- Eclipse
- Gnome
- KDE
- Qt
- Ruby on Rails
- Android
- PostgreSQL
- Debian
- X.org

## Download Git

The latest stable Git release is

**v1.7.7.1**

release notes (2011-10-23)

Windows | Mac OSX | Source

Older Releases
Git Source Repository

## Git Quick Start

### Cloning and Creating a Patch

```
$ git clone git://github.com/git/hello-world.git
$ cd hello-world
$ (edit files)
$ git add (files)
$ git commit -m 'Explain what I changed'
```

### Creating and Commiting

```
$ cd (project-directory)
$ git init
$ (add some files)
$ git add .
$ git commit -m 'Initial commit'
```

# git-scm.com


Home | About Git | Documentation | Download | Tools & Hosting

## Git is...

Git is a **free & open source, distributed version control system** designed to handle everything from small to very large projects with speed and efficiency.

**Every Git clone is a full-fledged repository** with complete history and full revision tracking capabilities, not dependent on network access or a central server. **Branching and merging are fast** and easy to do.

Git is used for version control of files, much like tools such as Mercurial, Bazaar, Subversion, CVS, Perforce, and Team Foundation Server.

## Projects using Git

- Git
- Linux Kernel
- Perl
- Eclipse
- Gnome
- KDE
- Qt
- Ruby on Rails
- Android
- PostgreSQL
- Debian
- X.org

## Download Git

The latest stable Git release is

**v1.7.7.1**

release notes (2011-10-23)

Windows | Mac OSX | Source

Older Releases
Git Source Repository

## Git Quick Start

### Cloning and Creating a Patch

```
$ git clone git://github.com/git/hello-world.git
$ cd hello-world
$ (edit files)
$ git add (files)
$ git commit -m 'Explain what I changed'
```

### Creating and Commiting

```
$ cd (project-directory)
$ git init
$ (add some files)
$ git add .
$ git commit -m 'Initial commit'
```

# git-scm.com

# Git Reference

Reference | About | § | Site Source


**Getting and Creating Projects**

- init
- clone

**Basic Snapshotting**

- add
- status
- diff
- commit
- reset
- rm, mv

**Branching and Merging**

- branch
- checkout
- merge
- log
- tag

**Sharing and Updating Projects**

- fetch, pull
- push
- remote

**Inspection and Comparison**


## INTRODUCTION TO THE GIT REFERENCE

This is the Git reference site. This is meant to be a quick reference for learning and remembering the most important and commonly used Git commands. The commands are organized into sections of the type of operation you may be trying to do, and will present the common options and commands needed to accomplish these common tasks.

Each section will link to the next section, so it can be used as a tutorial. Every page will also link to more in-depth Git documentation such as the offical manual pages and relevant sections in the **Pro Git book**, so you can learn more about any of the commands. First, we'll start with thinking about source code management like Git does.

## HOW TO THINK LIKE GIT

This first thing that is important to understand about Git is that it thinks about version control very differently than Subversion or Perforce or whatever SCM you may be used to. It is often easier to learn Git by trying to forget your assumptions about how version control works and try to think about it in the Git way.

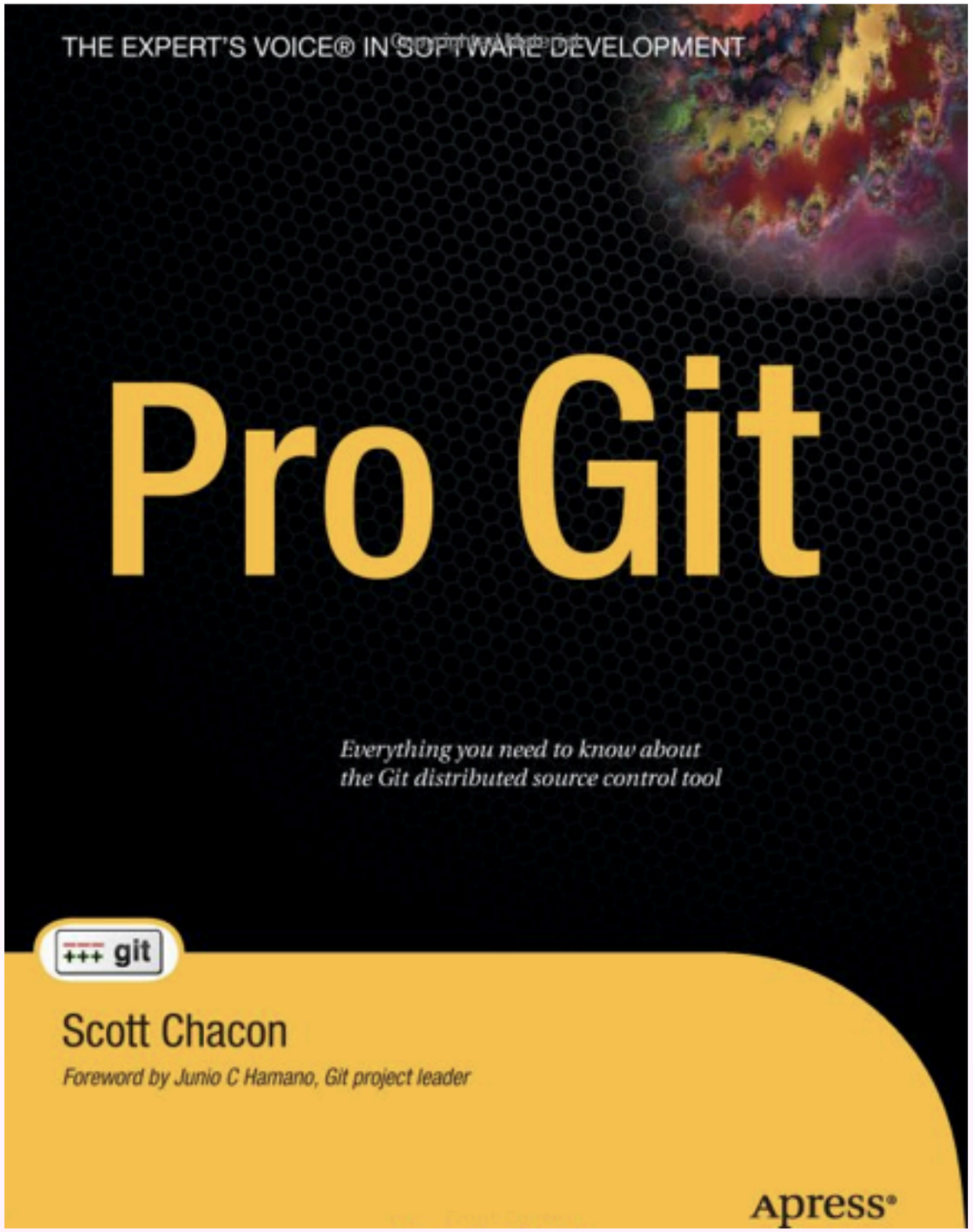
Let's start from scratch. Assume you are designing a new source code management system. How did you do basic version control before you used a tool for it? Chances are that you simply copied your project directory to save what it looked like at that point.

```
$ cp -R project project.bak
```

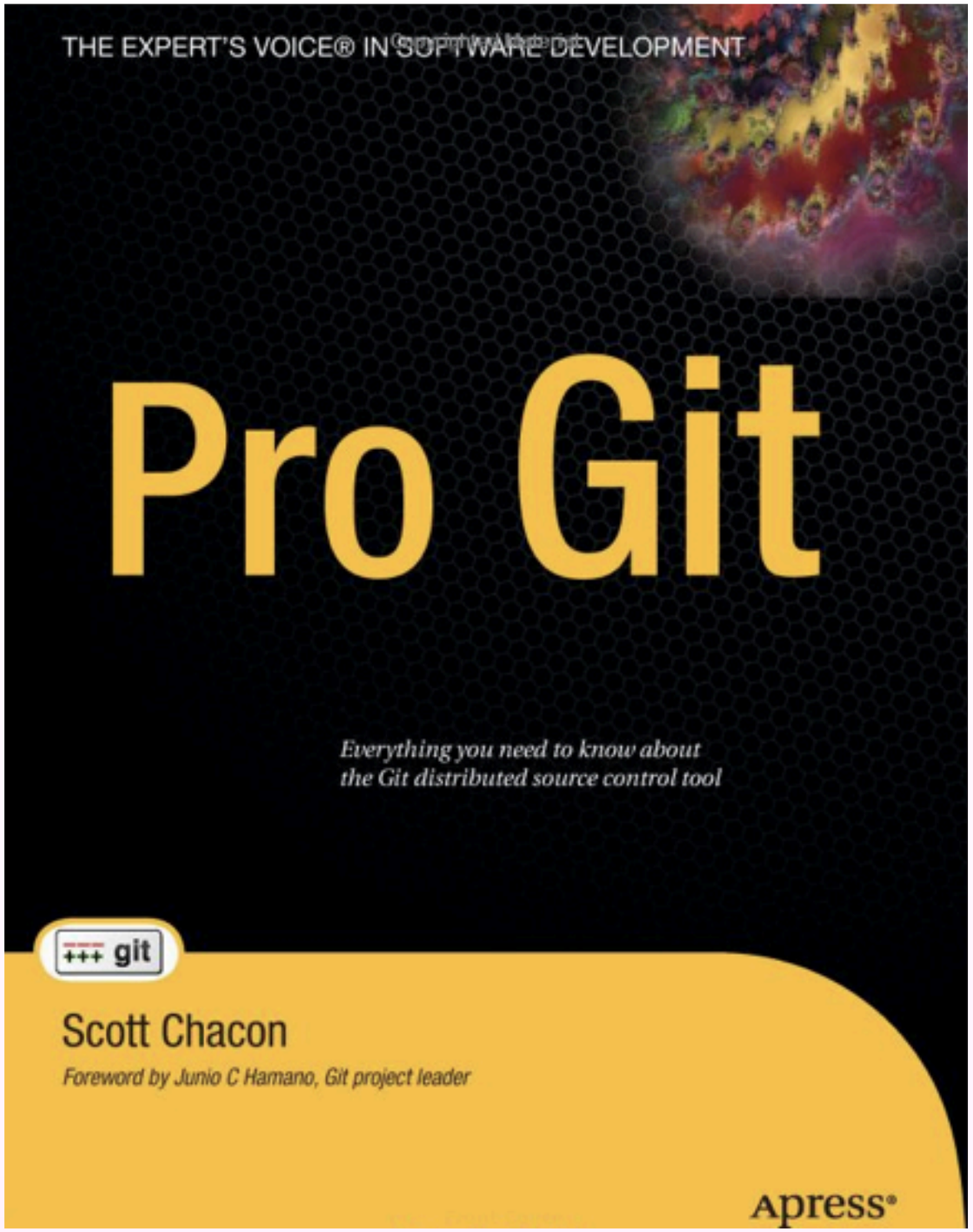
That way, you can easily revert files that get messed up later, or see what you have changed by comparing what the project looks like now to what it looked like when you copied it.

If you are really paranoid, you may do this often, maybe putting the date in the name of the backup:

gitref.org


THE EXPERT'S VOICE® IN SOFTWARE DEVELOPMENT
Pro Git
Everything you need to know about
the Git distributed source control tool
git
Scott Chacon
Foreword by Junio C Hamano, Git project leader
Apress®


progit.org


THE EXPERT'S VOICE® IN SOFTWARE DEVELOPMENT
Pro Git
Everything you need to know about
the Git distributed source control tool
+++ git
Scott Chacon
Foreword by Junio C Hamano, Git project leader
Apress®

progit.org

# Pro Git professional version control

Home Book Blog About Support Us GitHub Twitter

This is an in-progress translation.
To help translate the book, please fork the book at GitHub and push your contributions.

## 1. Empezando

1.1 - Acerca del control de versiones
1.2 - Una breve historia de Git
1.3 - Fundamentos de Git
1.4 - Instalando Git
1.5 - Configurando Git por primera vez
1.6 - Obteniendo ayuda
1.7 - Resumen


## 2. Fundamentos de Git

2.1 - Obteniendo un repositorio Git
2.2 - Guardando cambios en el repositorio
2.3 - Viendo el histórico de confirmaciones
2.4 - Deshaciendo cosas
2.5 - Trabajando con repositorios remotos
2.6 - Creando etiquetas
2.7 - Consejos y trucos
2.8 - Resumen

## 3. Ramificaciones en Git

3.1 - ¿Qué es una rama?
3.2 - Procedimientos básicos para ramificar y fusionar

Support this site by buying a print version of Pro Git

Follow the author on Twitter for updates and Git tips

**Also available in:**

progit.org/book/es

# recursos

# recursos

git-scm.com

# recursos

git-scm.com

gitref.org

# recursos

git-scm.com

gitref.org

progit.org

@chacon

</mi>

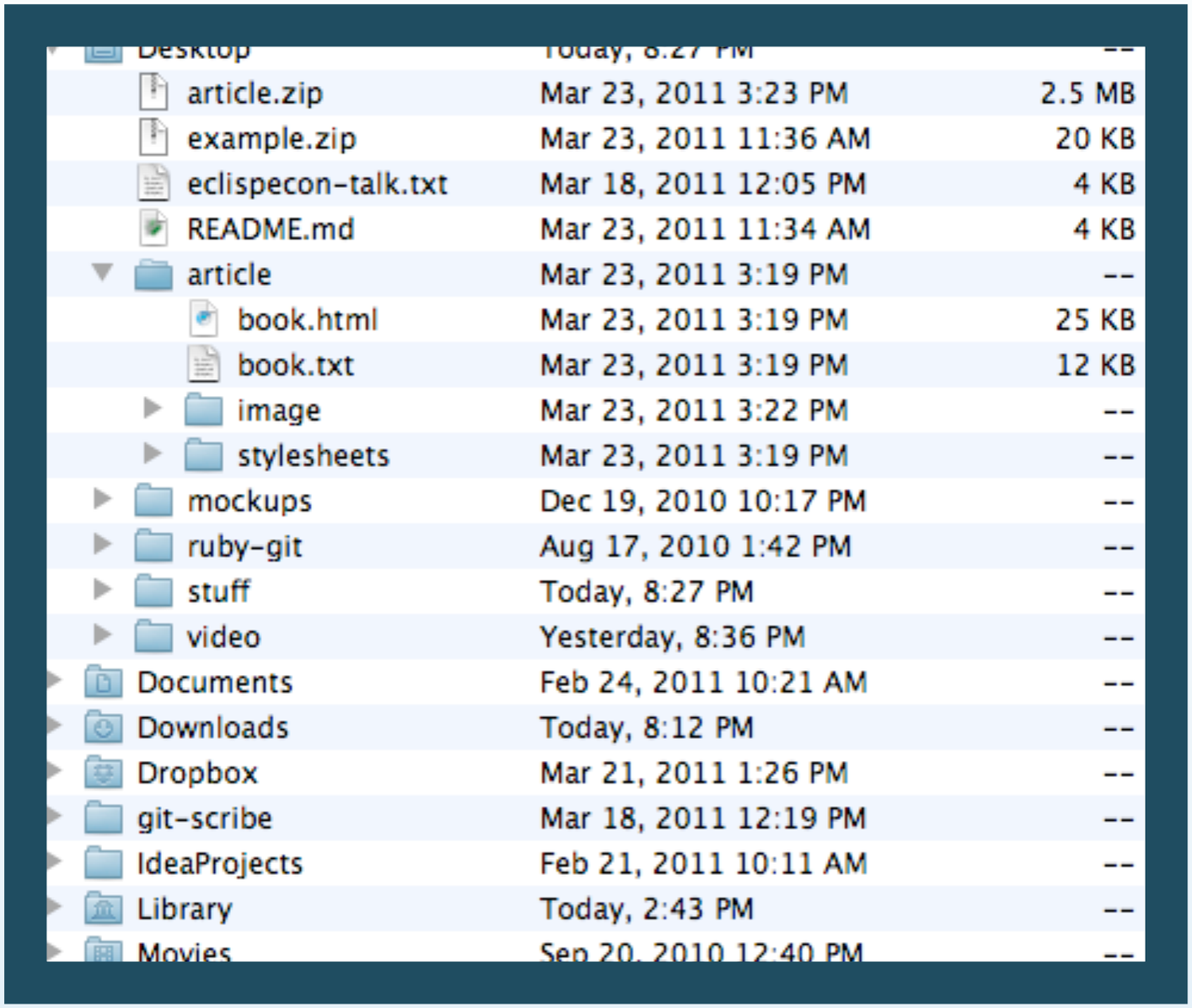
reintroducción a

# git

# manejar árboles

“árbol”

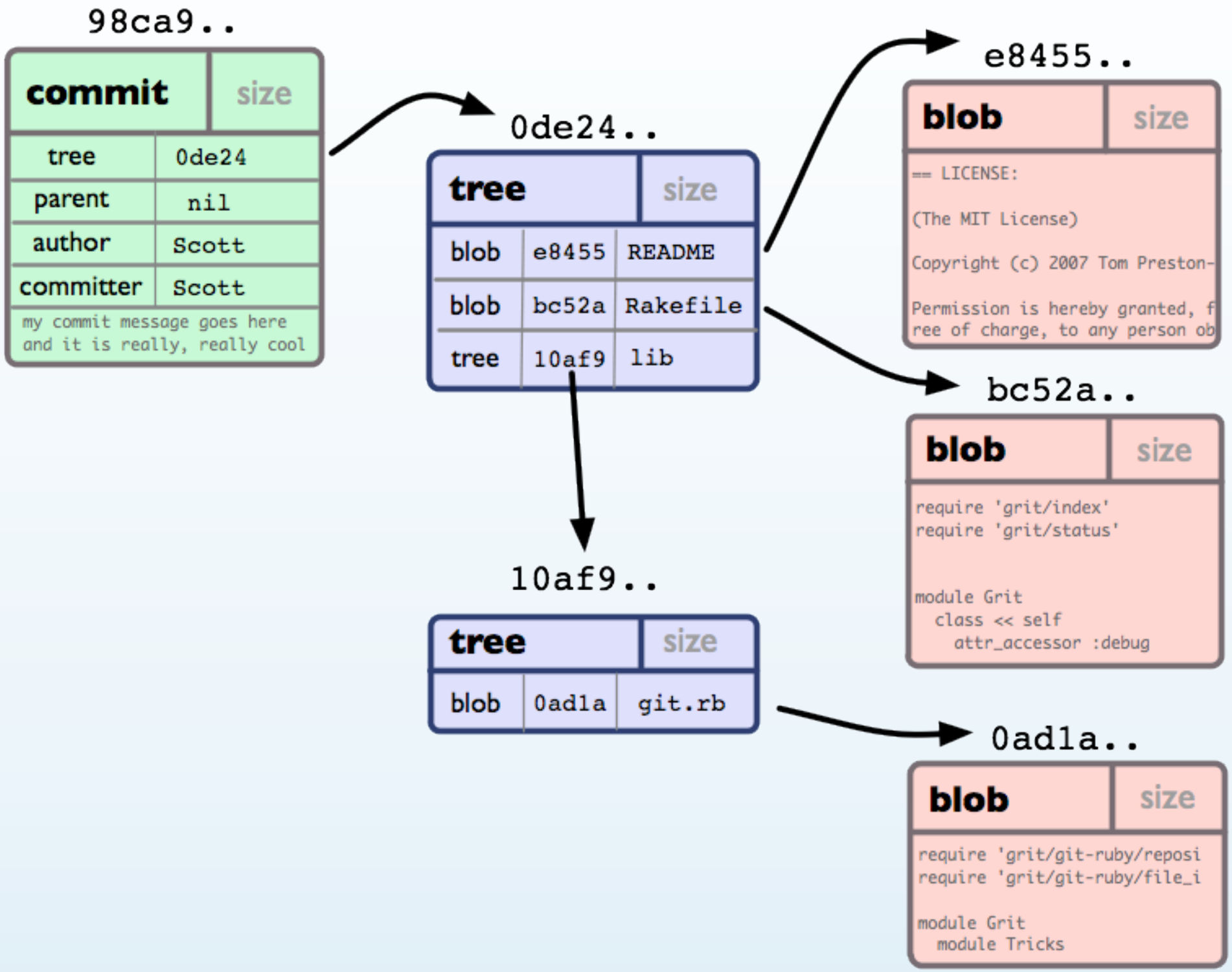
# árbol snapshot

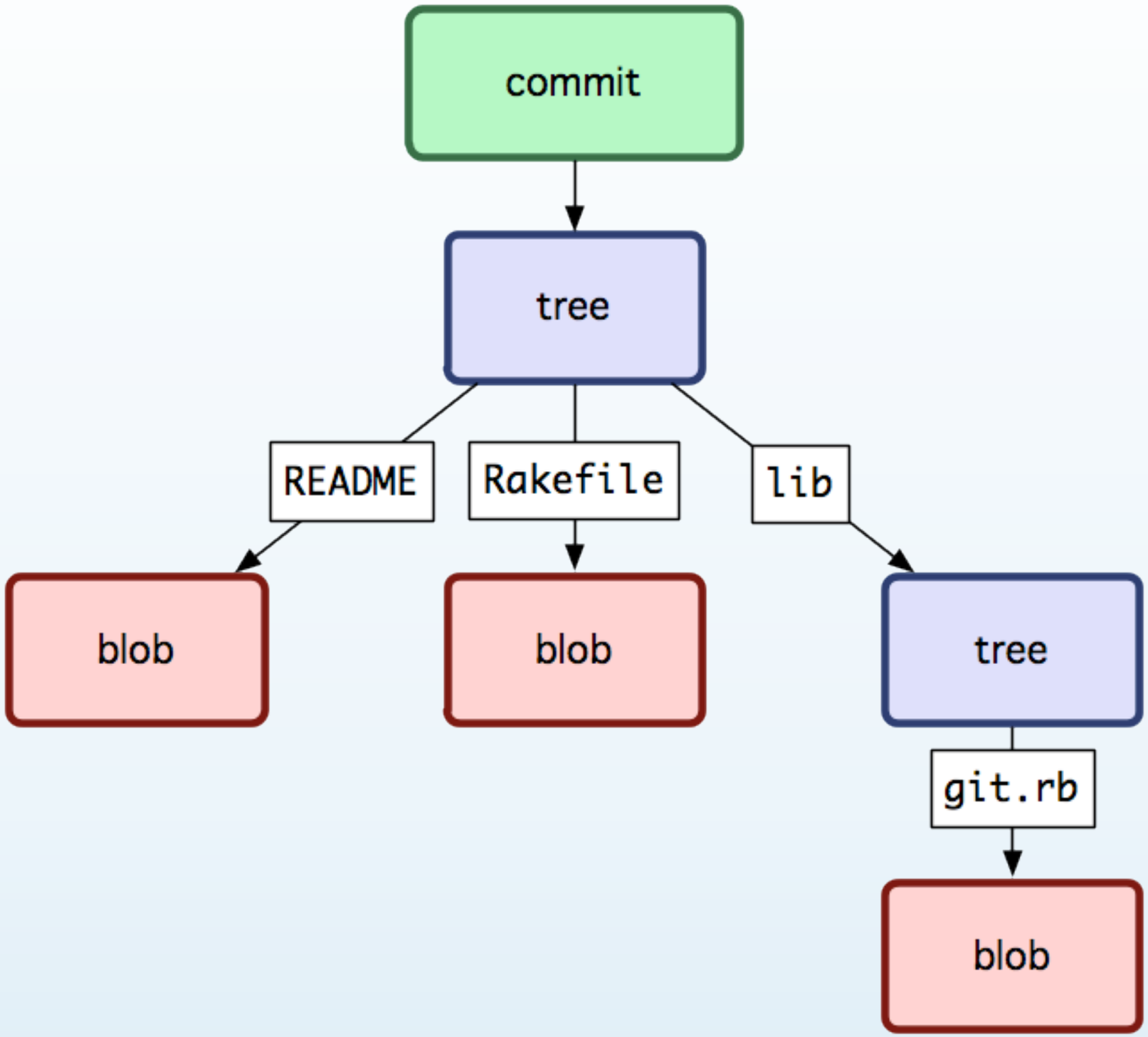

Desktop Today, 8:27 PM --
article.zip Mar 23, 2011 3:23 PM 2.5 MB
example.zip Mar 23, 2011 11:36 AM 20 KB
eclispecon-talk.txt Mar 18, 2011 12:05 PM 4 KB
README.md Mar 23, 2011 11:34 AM 4 KB
article Mar 23, 2011 3:19 PM --
book.html Mar 23, 2011 3:19 PM 25 KB
book.txt Mar 23, 2011 3:19 PM 12 KB
image Mar 23, 2011 3:22 PM --
stylesheets Mar 23, 2011 3:19 PM --
mockups Dec 19, 2010 10:17 PM --
ruby-git Aug 17, 2010 1:42 PM --
stuff Today, 8:27 PM --
video Yesterday, 8:36 PM --
Documents Feb 24, 2011 10:21 AM --
Downloads Today, 8:12 PM --
Dropbox Mar 21, 2011 1:26 PM --
git-scribe Mar 18, 2011 12:19 PM --
IdeaProjects Feb 21, 2011 10:11 AM --
Library Today, 2:43 PM --
Movies Sep 20, 2010 12:40 PM --

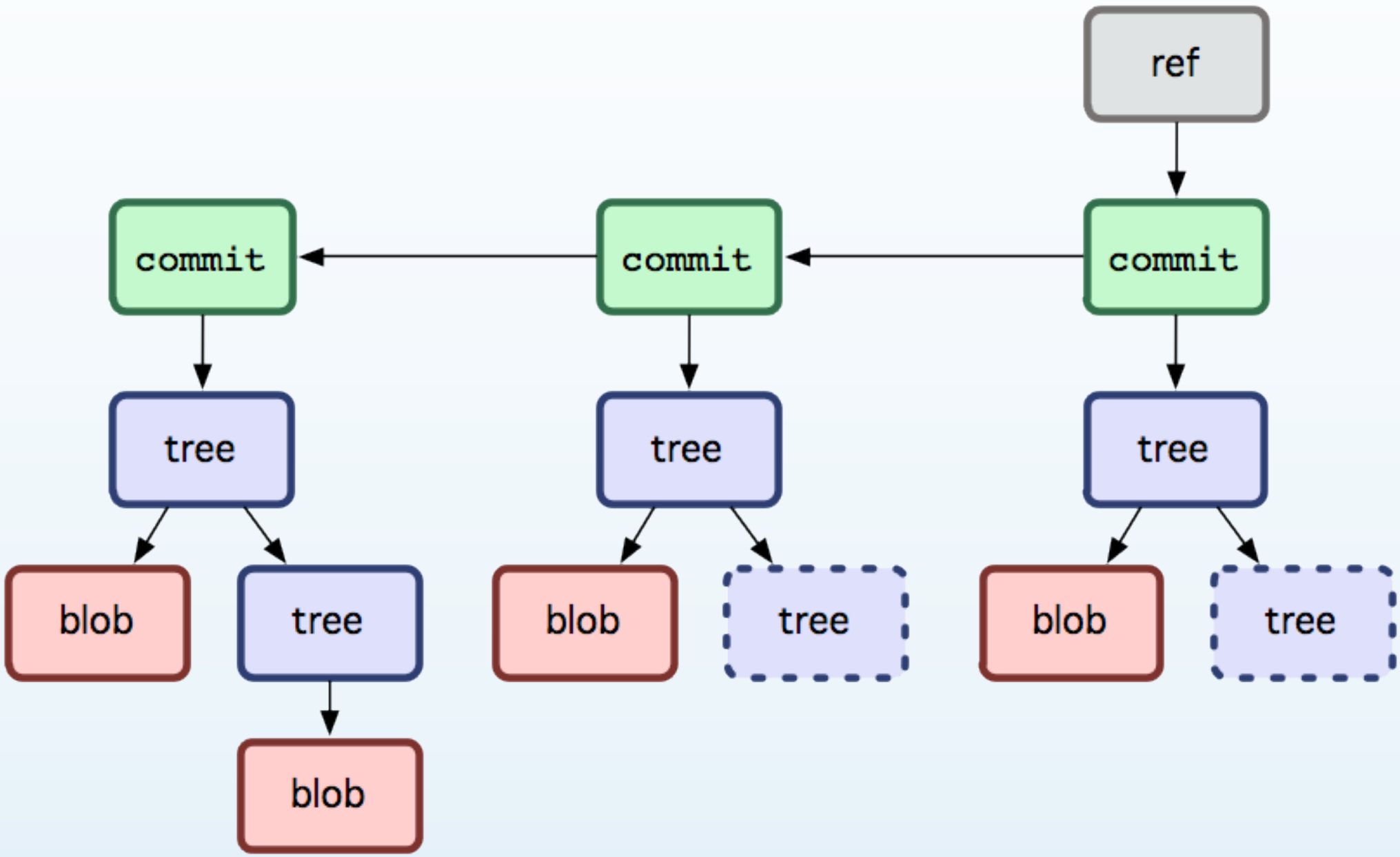
# ejemplo

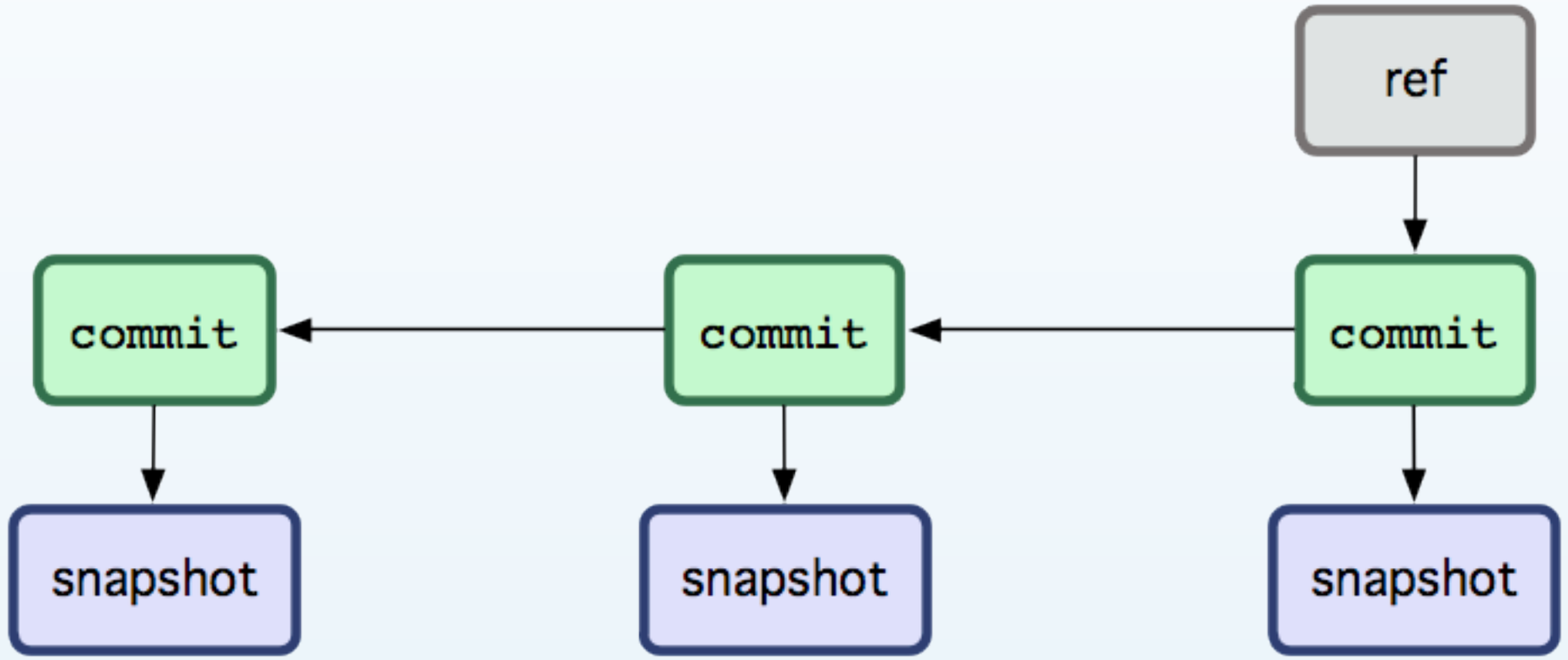
```
$ tree
.
├── README
├── Rakefile
└── lib
    └── git.rb

1 directory, 3 files
```


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
blob
bc52a
Rakefile
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
bc52a..
blob
size
require 'grit/index'
require 'grit/status'
module Grit
class << self
attr_accessor :debug
10af9..
tree
size
blob
0ad1a
git.rb
0ad1a..
blob
size
require 'grit/git-ruby/reposi
require 'grit/git-ruby/file_i
module Grit
module Tricks


commit
tree
README
Rakefile
lib
blob
blob
tree
git.rb
blob


ref
commit
commit
commit
tree
tree
tree
blob
tree
blob
tree
blob
tree
blob
tree


ref
commit
commit
commit
snapshot
snapshot
snapshot

# git

# git

*commit* crea
árboles permanentes

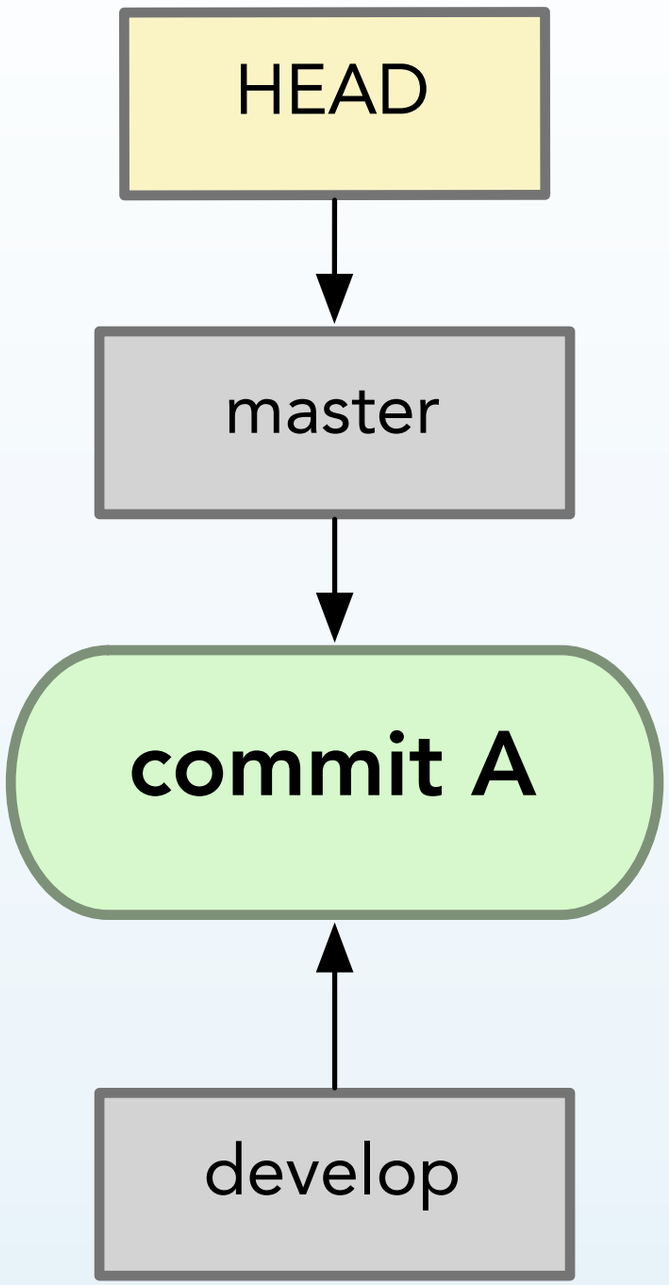
# PRIMER ACTO

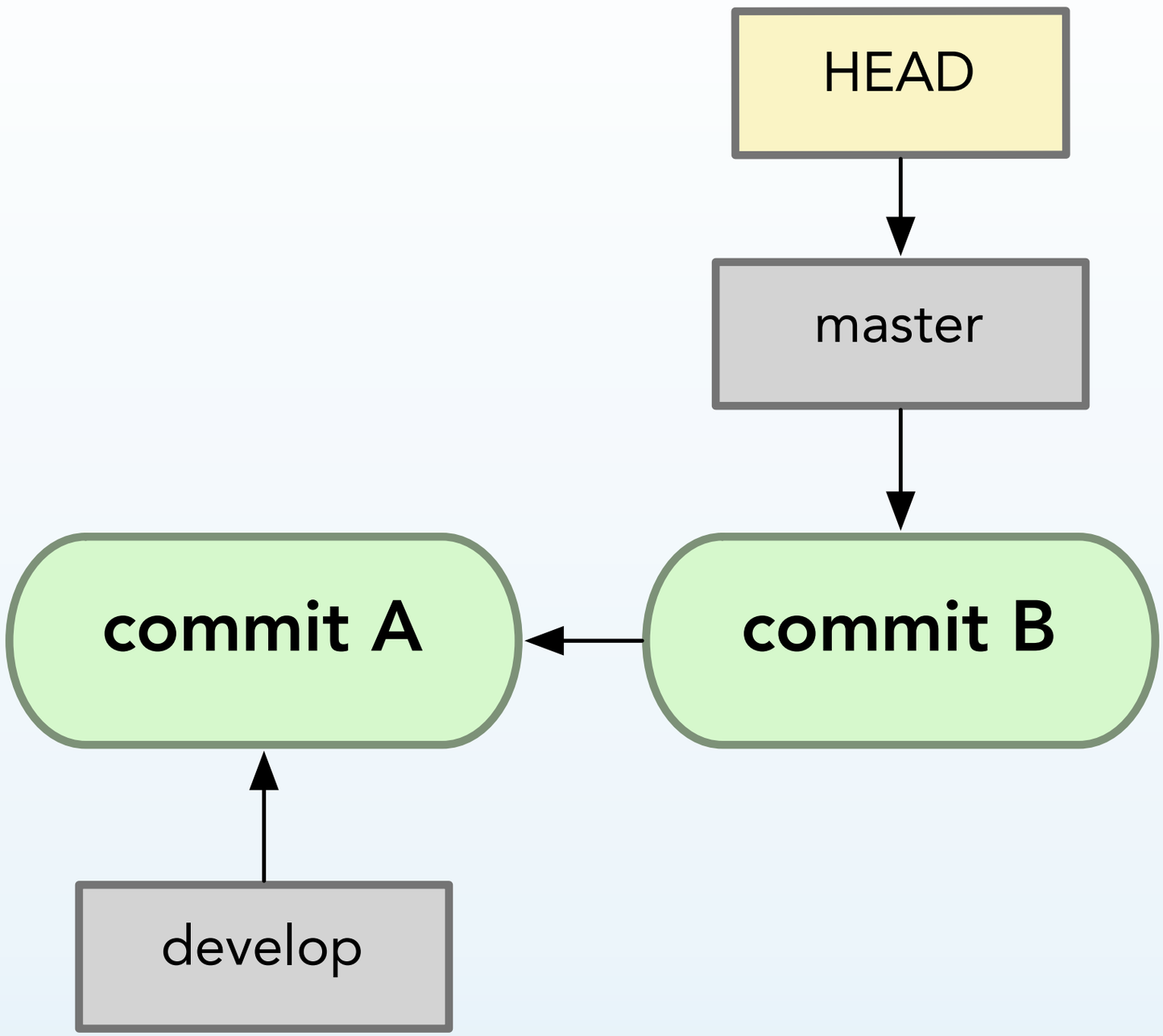
## los tres árboles

el primer árbol

# el HEAD

# ¿usan HEAD?


HEAD
master
commit A
develop


HEAD
master
commit A
commit B
develop

# HEAD

el último commit

el segundo árbol

# el ÍNDICE

# área de ensayo

# ¿un árbol?

# un árbol?

mas o menos

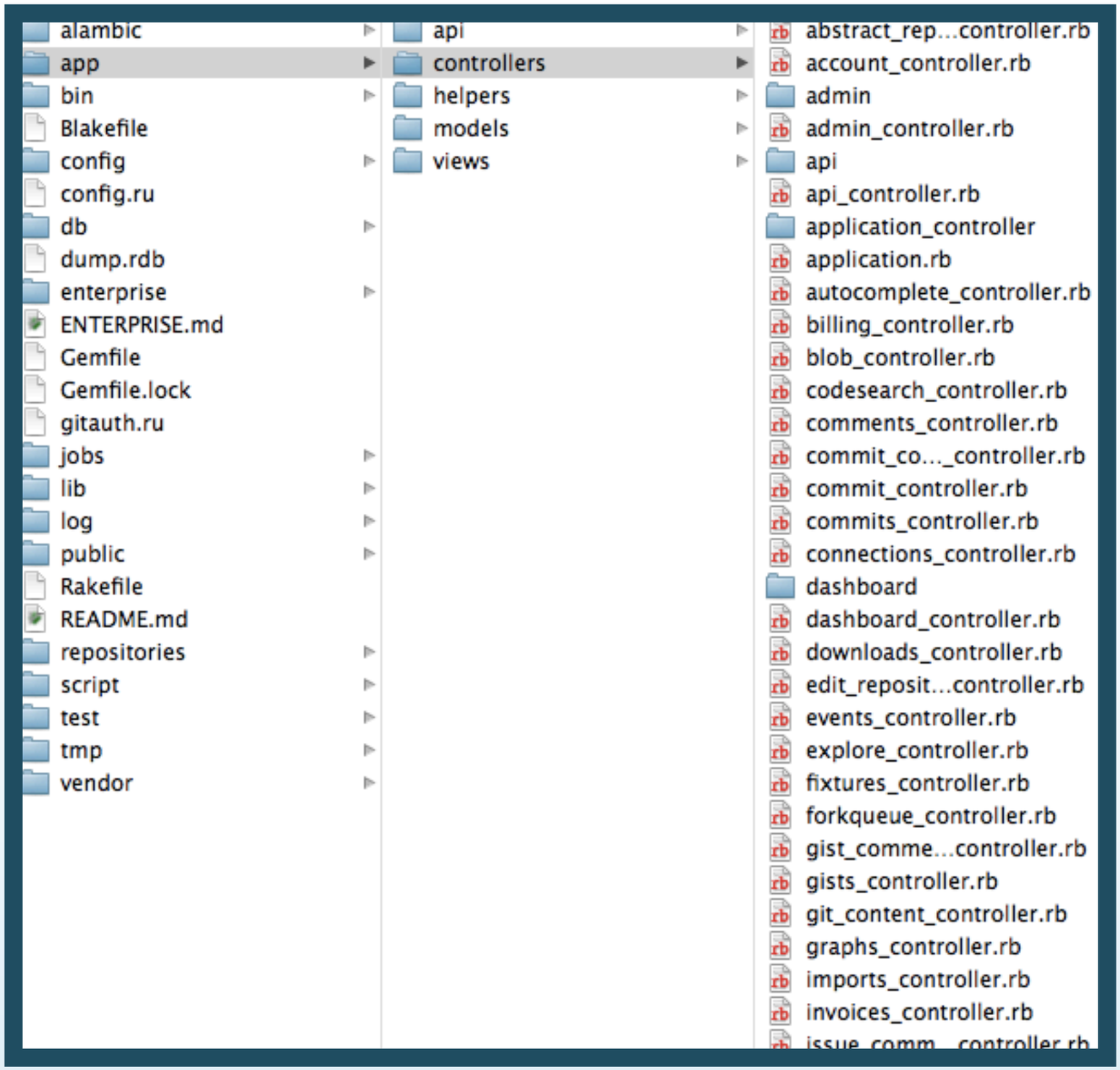
# índice

# índice

el proximo commit

el tercer árbol

# el directorio de trabajo


alambic
app
bin
Blakefile
config
config.ru
db
dump.rdb
enterprise
ENTERPRISE.md
Gemfile
Gemfile.lock
gitauth.ru
jobs
lib
log
public
Rakefile
README.md
repositories
script
test
tmp
vendor
api
controllers
helpers
models
views
abstract_rep...controller.rb
account_controller.rb
admin
admin_controller.rb
api
api_controller.rb
application_controller
application.rb
autocomplete_controller.rb
billing_controller.rb
blob_controller.rb
codesearch_controller.rb
comments_controller.rb
commit_co..._controller.rb
commit_controller.rb
commits_controller.rb
connections_controller.rb
dashboard
dashboard_controller.rb
downloads_controller.rb
edit_reposit...controller.rb
events_controller.rb
explore_controller.rb
fixtures_controller.rb
forkqueue_controller.rb
gist_comme...controller.rb
gists_controller.rb
git_content_controller.rb
graphs_controller.rb
imports_controller.rb
invoices_controller.rb

## HEAD

```
$ git ls-tree -r HEAD
100644 blob ad47ff7d328ff27e50bfcd0fb22baa1d680109bb  README
100644 blob 5de1607ba20b3ed555e8f77465d668005d545159  example.rb
100644 blob 6478a3173fd423085ce7685a37ec7fcf49401737  kidgloves.rb
```

## Índice

```
$ git ls-files -s
100644 ad47ff7d328ff27e50bfcd0fb22baa1d680109bb 0  README
100644 5de1607ba20b3ed555e8f77465d668005d545159 0  example.rb
100644 6478a3173fd423085ce7685a37ec7fcf49401737 0  kidgloves.rb
```

## Directorio de Trabajo

```
$ ls -l
-rw-r--r--  1 schacon  staff   610 Sep 26 09:38 README
-rw-r--r--  1 schacon  staff   209 Sep 26 09:38 example.rb
-rw-r--r--  1 schacon  staff  5024 Sep 26 09:38 kidgloves.rb
```

## HEAD

```
$ git ls-tree -r HEAD
100644 blob ad47ff7d328ff27e50bfcd0fb22baa1d680109bb  README
100644 blob 5de1607ba20b3ed555e8f77465d668005d545159  example.rb
100644 blob 6478a3173fd423085ce7685a37ec7fcf49401737  kidgloves.rb
```

## Índice

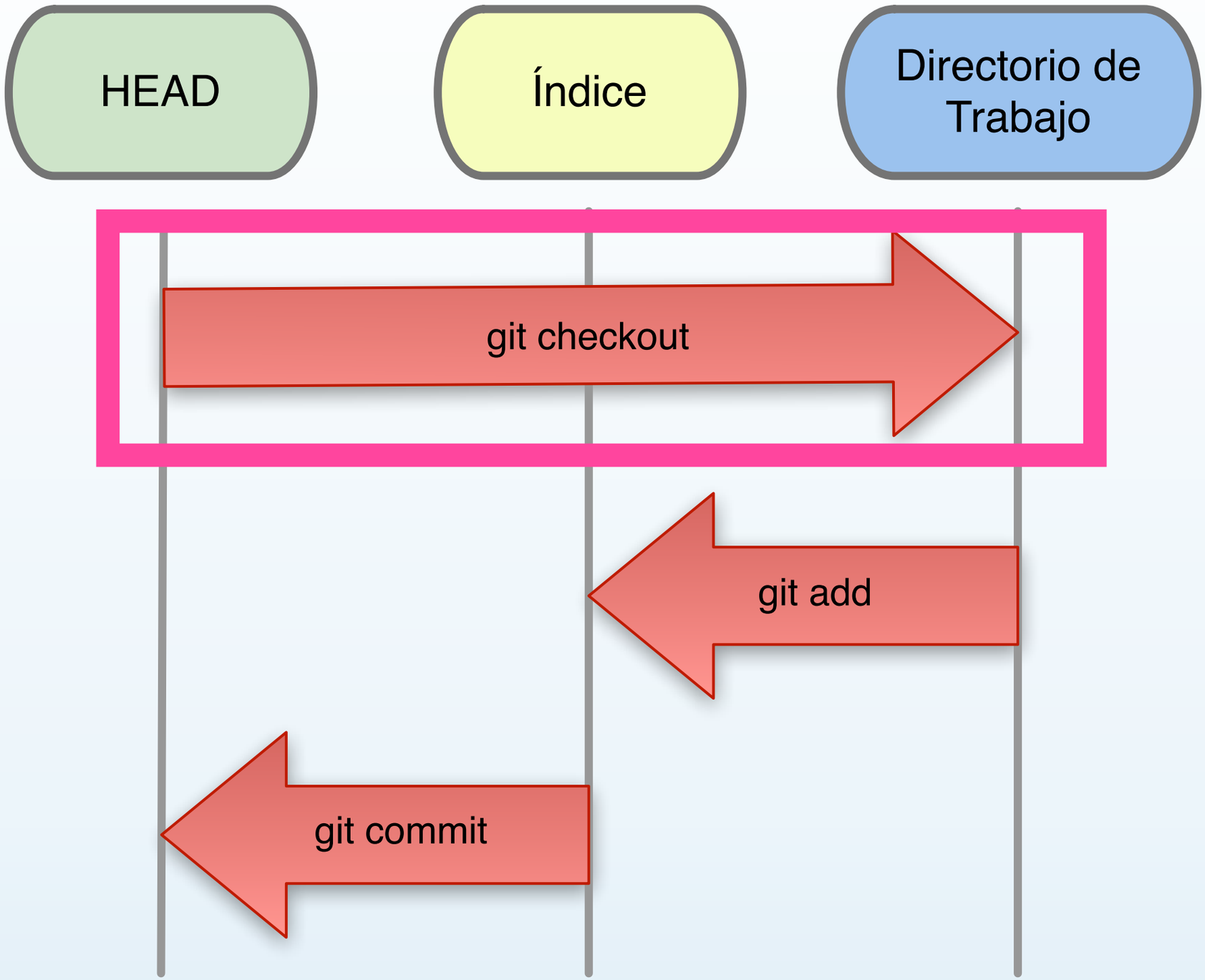
```
$ git ls-files -s
100644 ad47ff7d328ff27e50bfcd0fb22baa1d680109bb 0  README
100644 5de1607ba20b3ed555e8f77465d668005d545159 0  example.rb
100644 6478a3173fd423085ce7685a37ec7fcf49401737 0  kidgloves.rb
```

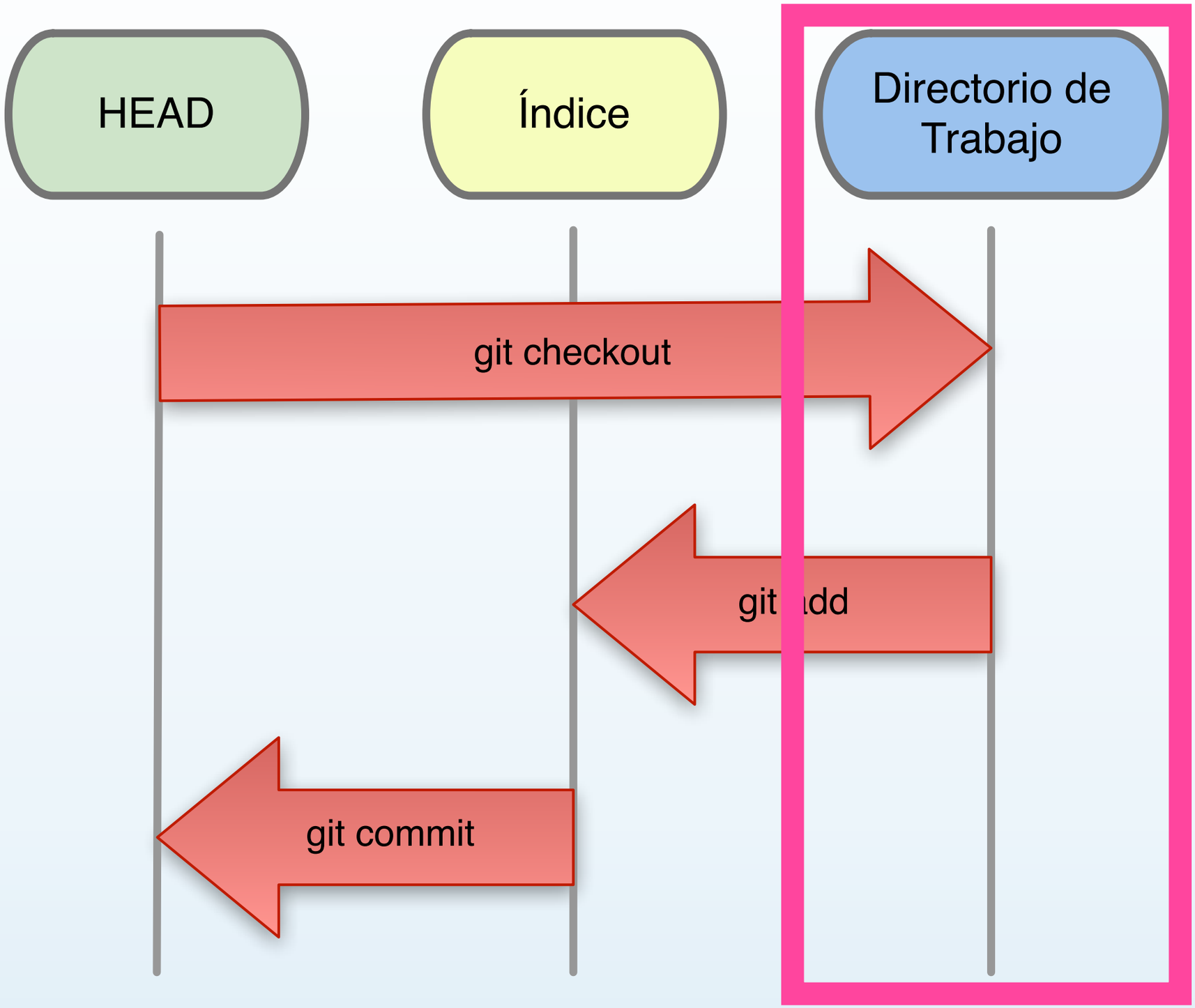
## Directorio de Trabajo

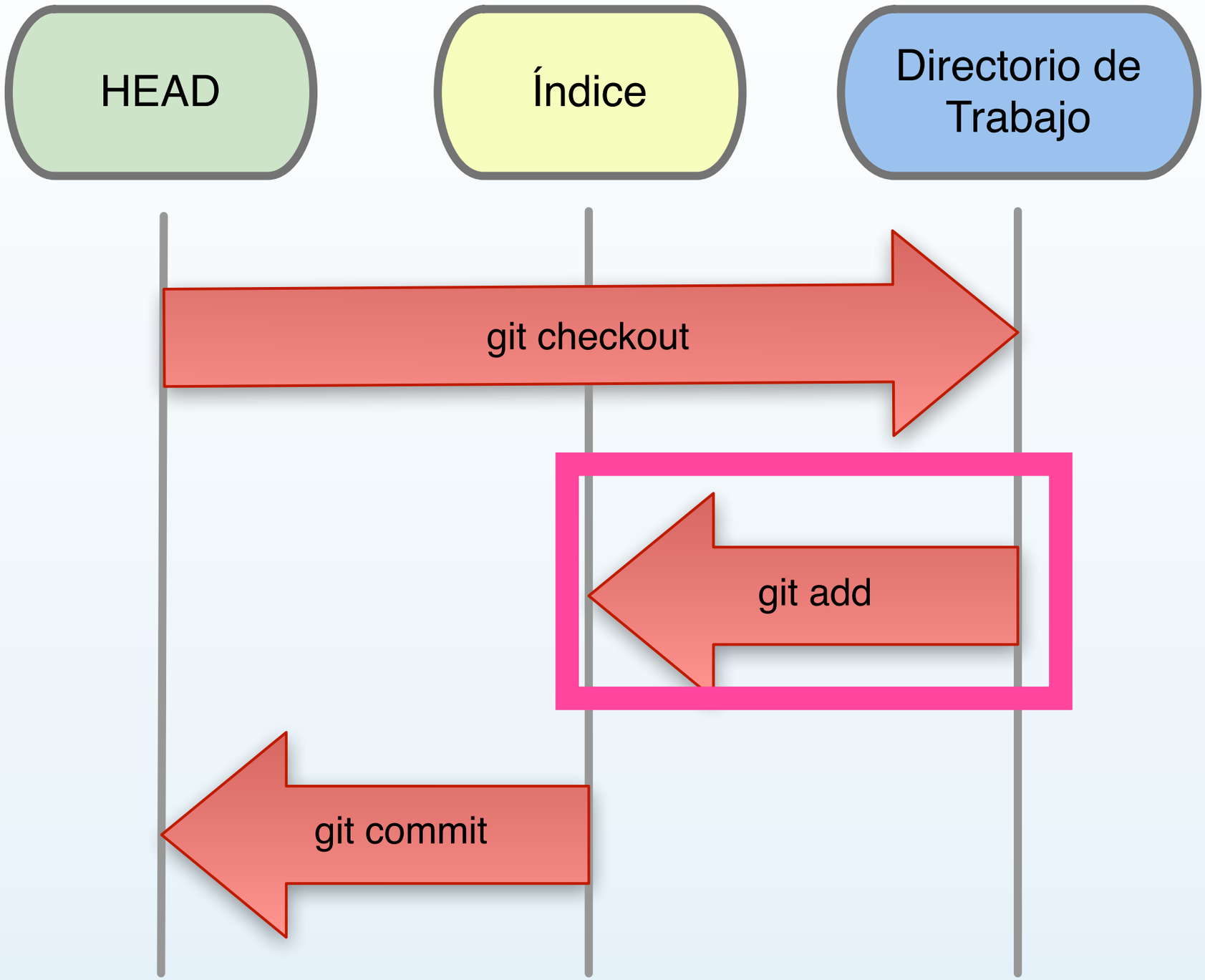
```
$ ls -l
-rw-r--r--  1 schacon  staff   610 Sep 26 09:38 README
-rw-r--r--  1 schacon  staff   209 Sep 26 09:38 example.rb
-rw-r--r--  1 schacon  staff  5024 Sep 26 09:38 kidgloves.rb
```

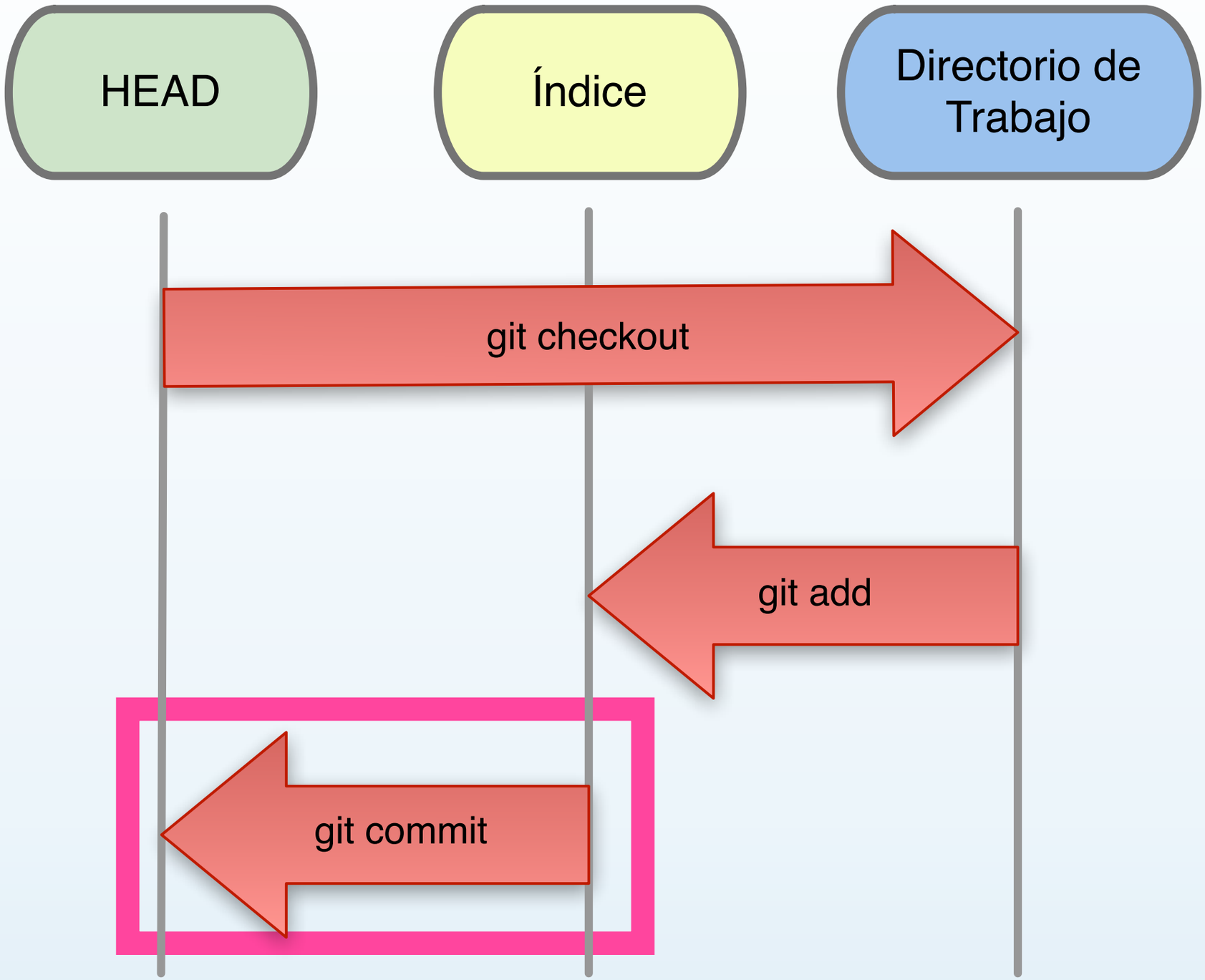
# TRES ÁRBOLES

## HEAD, Índice y Directorio de Trabajo


HEAD
Índice
Directorio de Trabajo
git checkout
git add
git commit


HEAD
Índice
Directorio de Trabajo
git checkout
git add
git commit


HEAD
Índice
Directorio de Trabajo
git checkout
git add
git commit


HEAD
Índice
Directorio de Trabajo
git checkout
git add
git commit

# funciones de los árboles

# funciones de los árboles

**HEAD** el último commit, generador del proximo

# funciones de los árboles

**HEAD** el último commit, generador del proximo

**Índice** el proximo commit

# funciones de los árboles

**HEAD** el último commit, generador del proximo

**Índice** el proximo commit

**Dir de Trabajo** caja de arena

# SEGUNDO ACTO

## trabajando con árboles

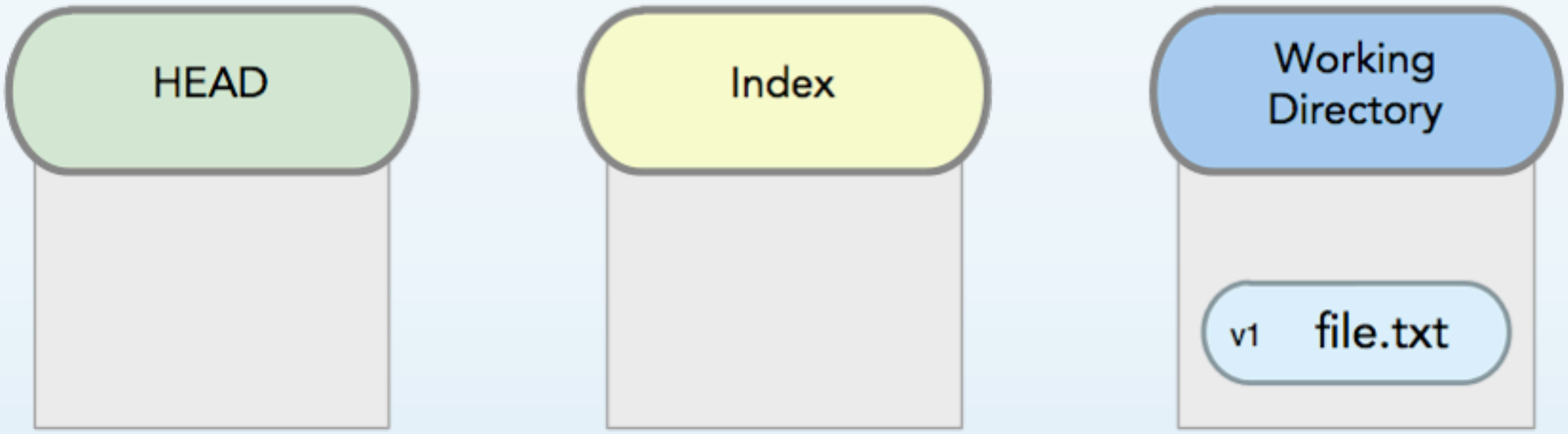
# git status

```
$ git status
# On branch master
# Your branch is behind 'origin/master' by 2 commits,
#  and can be fast-forwarded.
#
# Changes to be committed:
#   (use "git reset HEAD ..." to unstage)
#
#       modified:   jobs/email_reply.rb
#
# Changed but not updated:
#   (use "git add ..." to update what will be committed)
#   (use "git checkout -- ..." to discard changes
#     in working directory)
#
#       modified:   app/helpers/users_helper.rb
#       modified:   test/unit/email_reply_job_test.rb
#
```

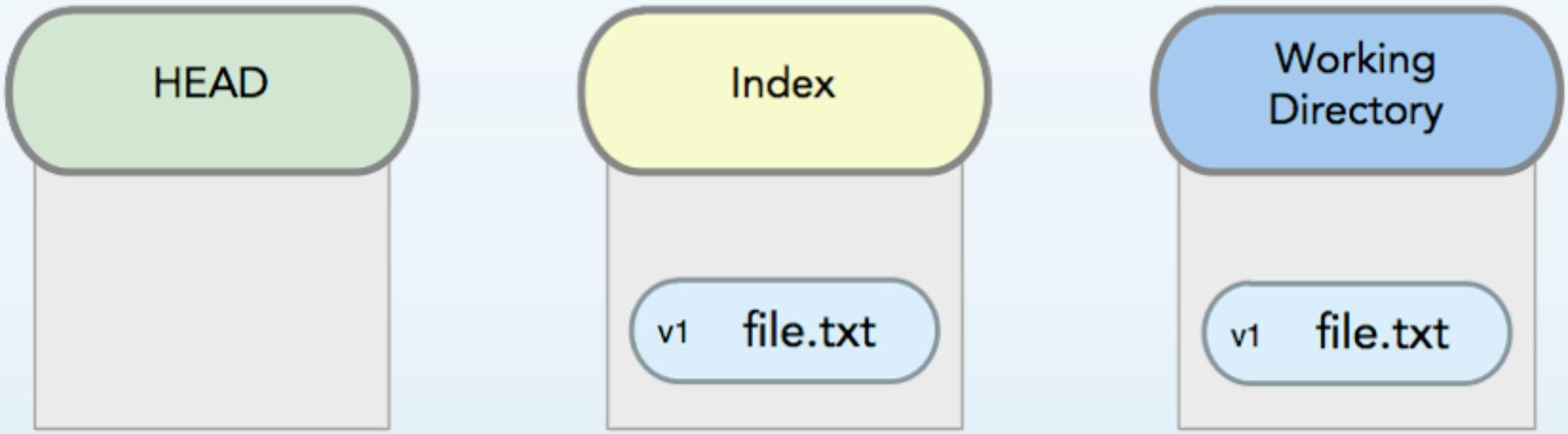
```
$ git status
# On branch master
# Your branch is behind 'origin/master' by 2 commits,
#  and can be fast-forwarded.
#
# Changes to be committed:
#   HEAD and index differ
#
#       modified:   jobs/email_reply.rb
#
# Changed but not updated:
#   index and working directory differ
#
#
#
#
#       modified:   app/helpers/users_helper.rb
#       modified:   test/unit/email_reply_job_test.rb
#
```

# ejemplo


HEAD
master
?
Git Repository

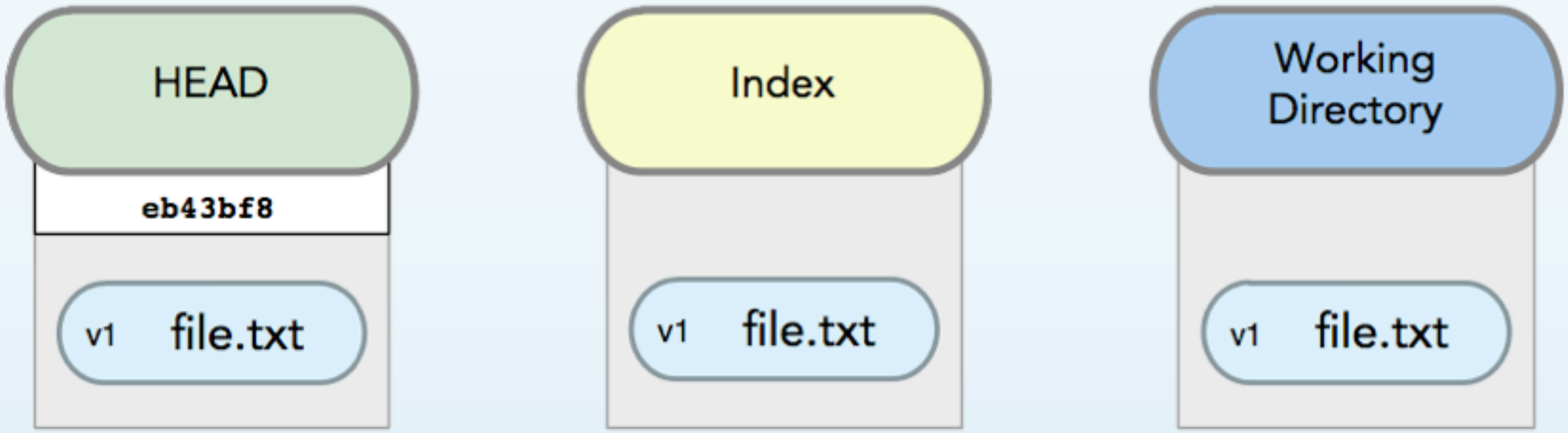

HEAD
Index
Working Directory
v1
file.txt


HEAD
master
?
Git Repository


HEAD
Index
v1 file.txt
Working Directory
v1 file.txt

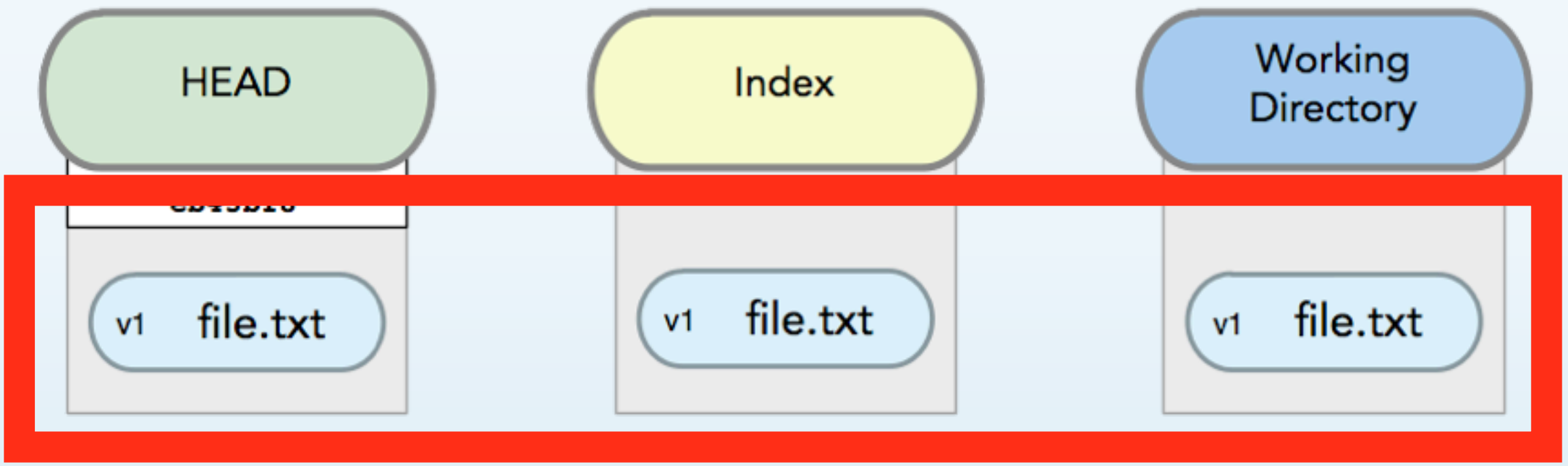
git add


HEAD
master
eb43bf8
v1 file.txt
Git Repository

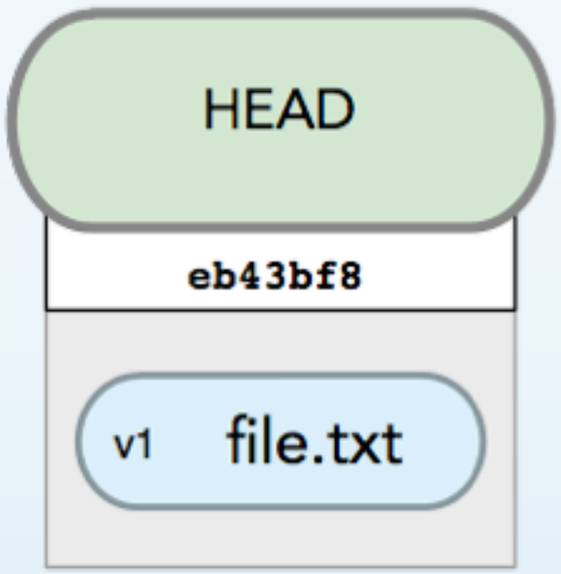

HEAD
eb43bf8
v1 file.txt
Index
v1 file.txt
Working Directory
v1 file.txt

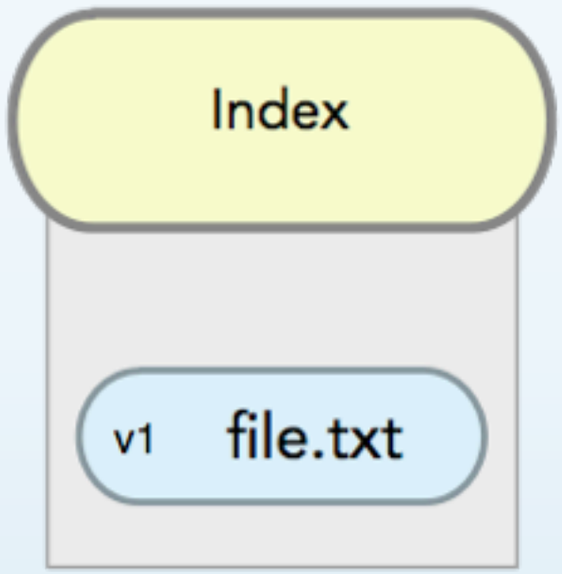
# git commit

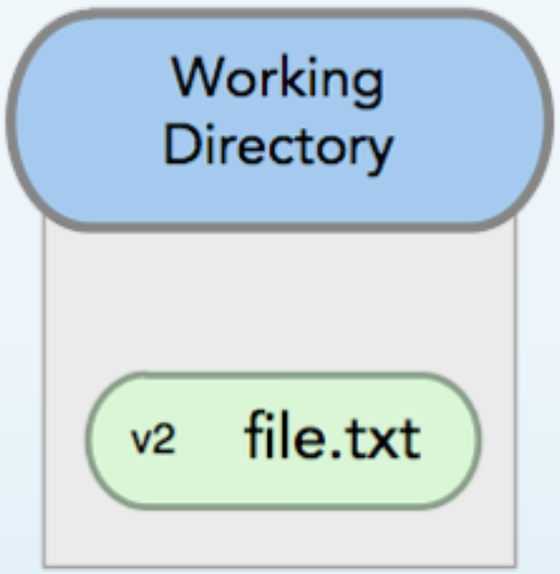

HEAD
master
eb43bf8
v1 file.txt
Git Repository


HEAD
Index
Working Directory
v1 file.txt
v1 file.txt
v1 file.txt


HEAD
master
eb43bf8
v1 file.txt
Git Repository


HEAD
eb43bf8
v1 file.txt

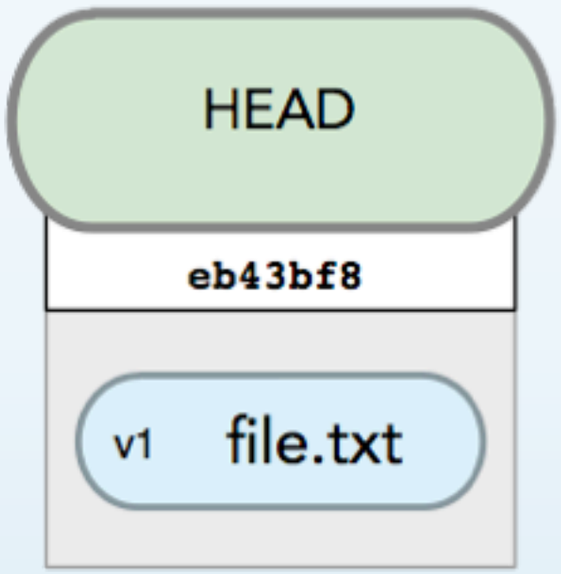

Index
v1 file.txt

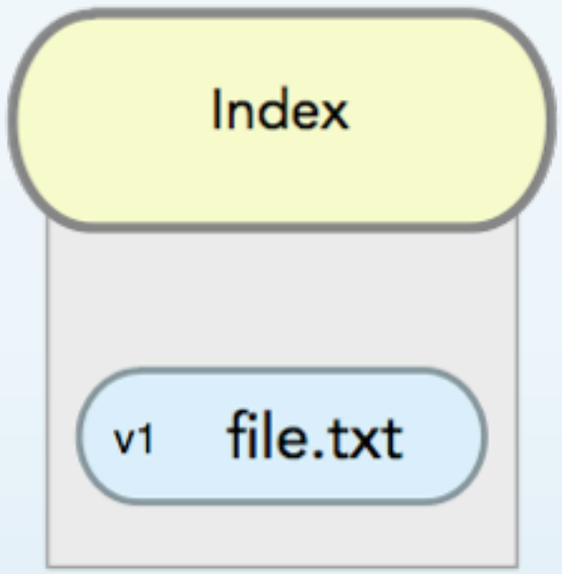

Working Directory
v2 file.txt

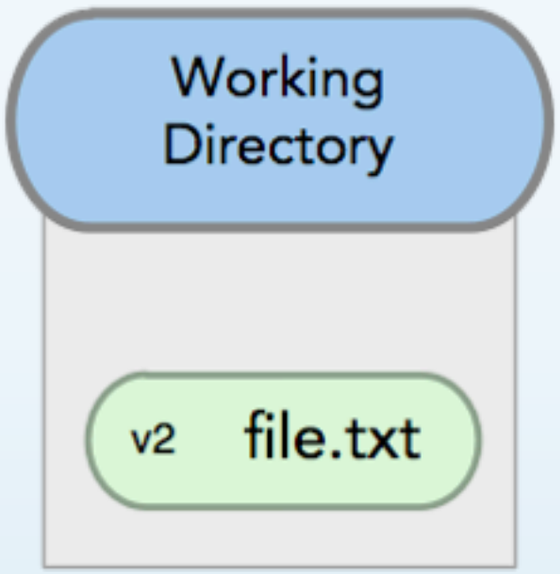
edit file

```
$ git status
# On branch master
# Your branch is behind 'origin/mas
#  and can be fast-forwarded.
#
# Changed but not updated:
#    (use "git add ..." to update wh
#    (use "git checkout -- ..." to d
#      in working directory)
#
#        modified:   file.txt
#
```


HEAD
master
eb43bf8
v1 file.txt
Git Repository

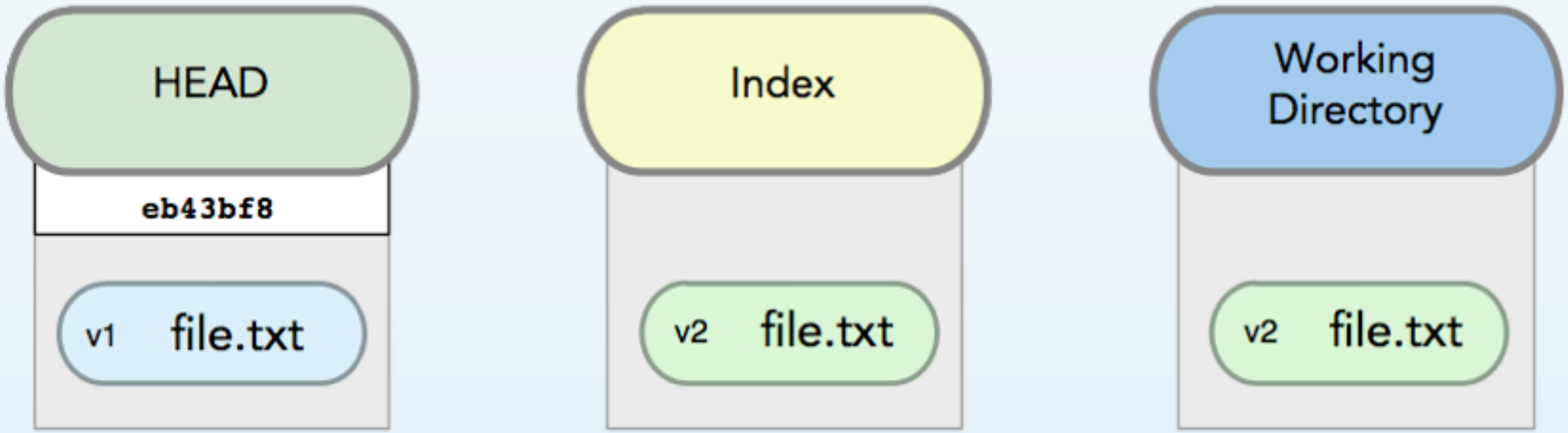

HEAD
eb43bf8
v1 file.txt


Index
v1 file.txt


Working Directory
v2 file.txt

edit file

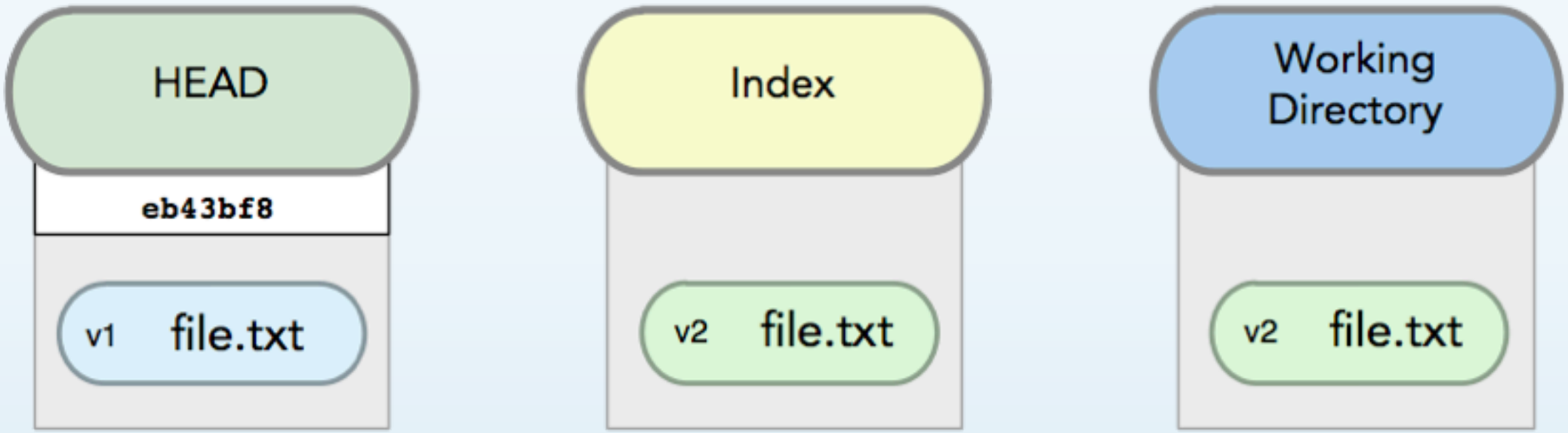

HEAD
master
eb43bf8
v1 file.txt
Git Repository


HEAD
eb43bf8
v1 file.txt
Index
v2 file.txt
Working Directory
v2 file.txt

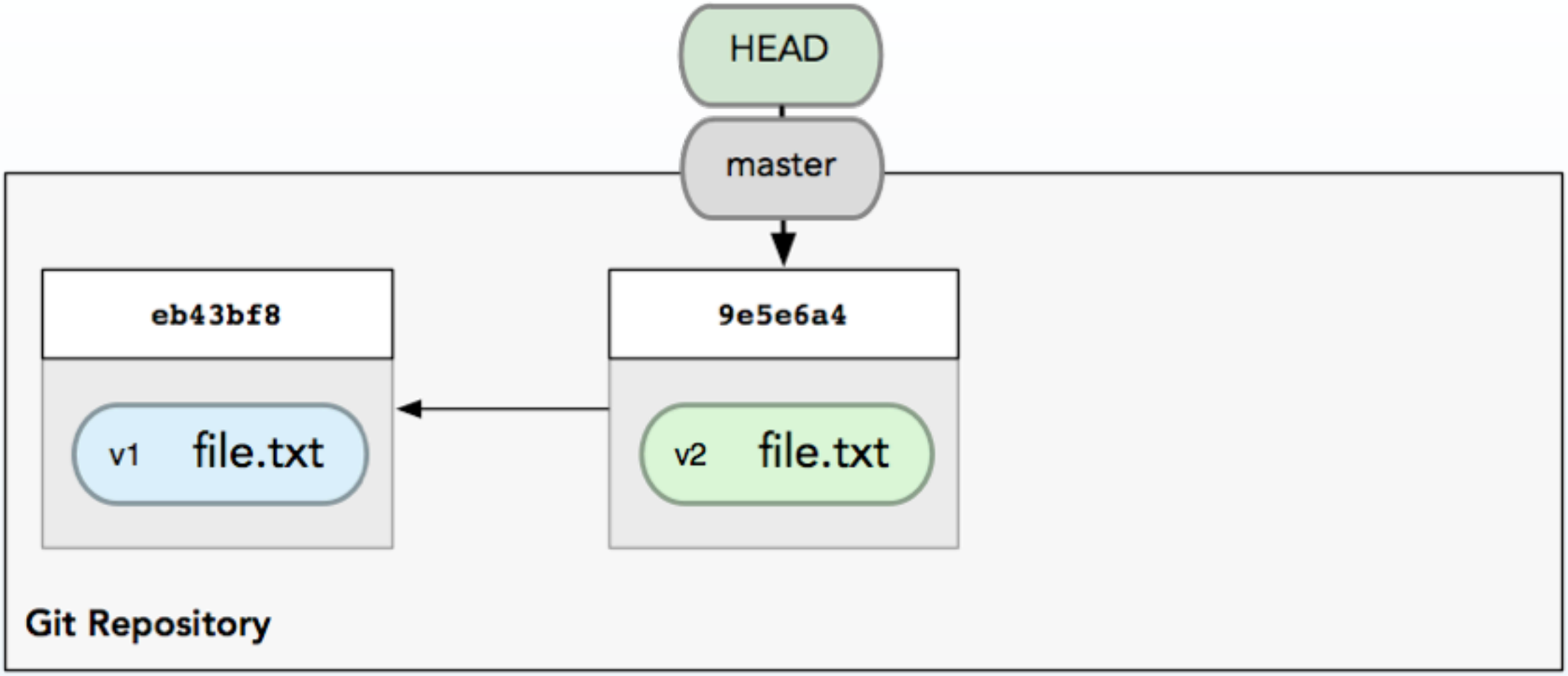
git add

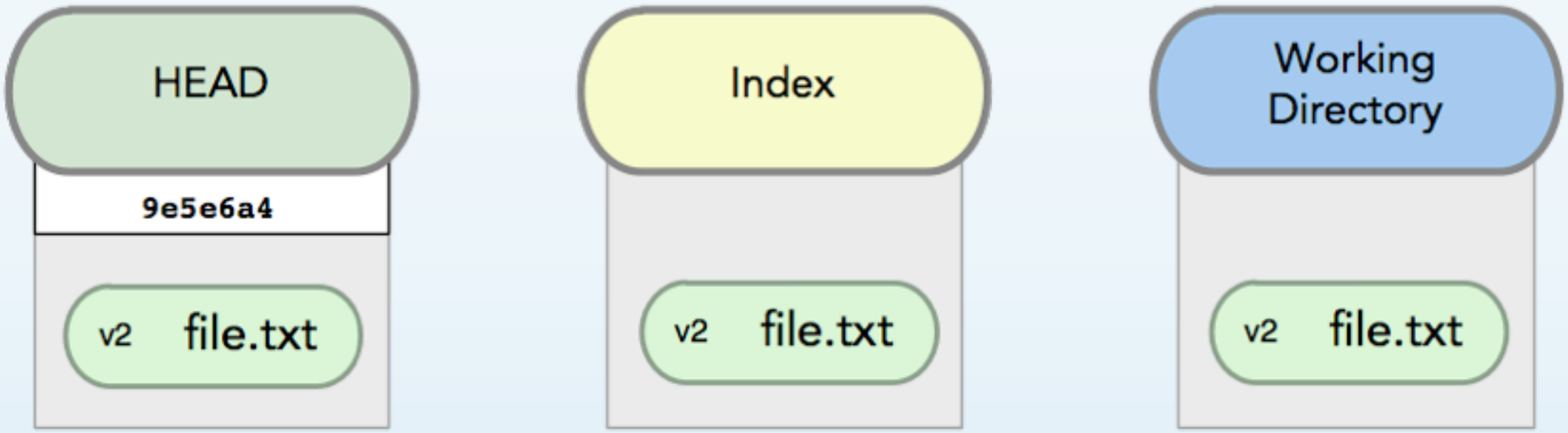
```
$ git status
# On branch master
# Your branch is behind 'origin/m
#  and can be fast-forwarded.
#
# Changes to be committed:
#   (use "git reset HEAD ..." to
#
#       modified:   file.txt
#
```


HEAD
master
eb43bf8
v1 file.txt
Git Repository

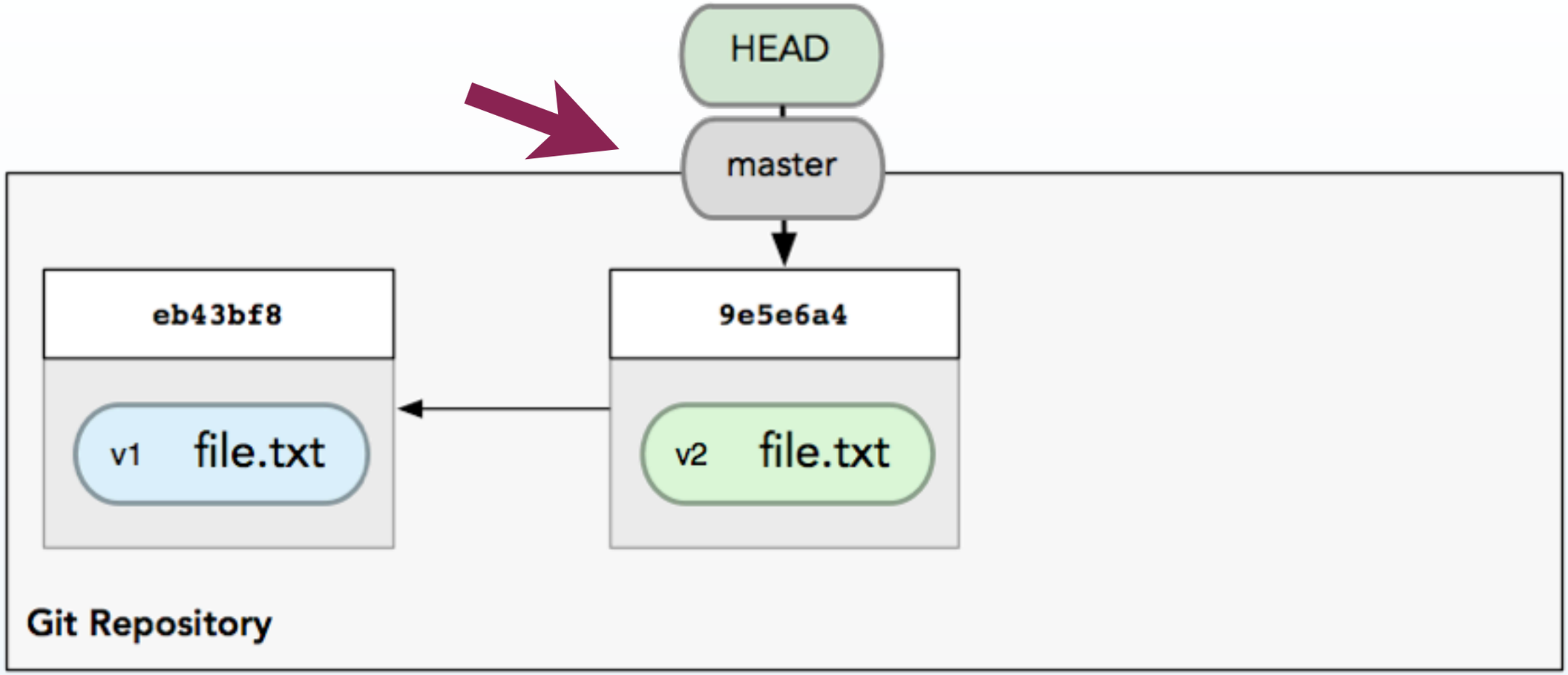

HEAD
eb43bf8
v1 file.txt
Index
v2 file.txt
Working Directory
v2 file.txt

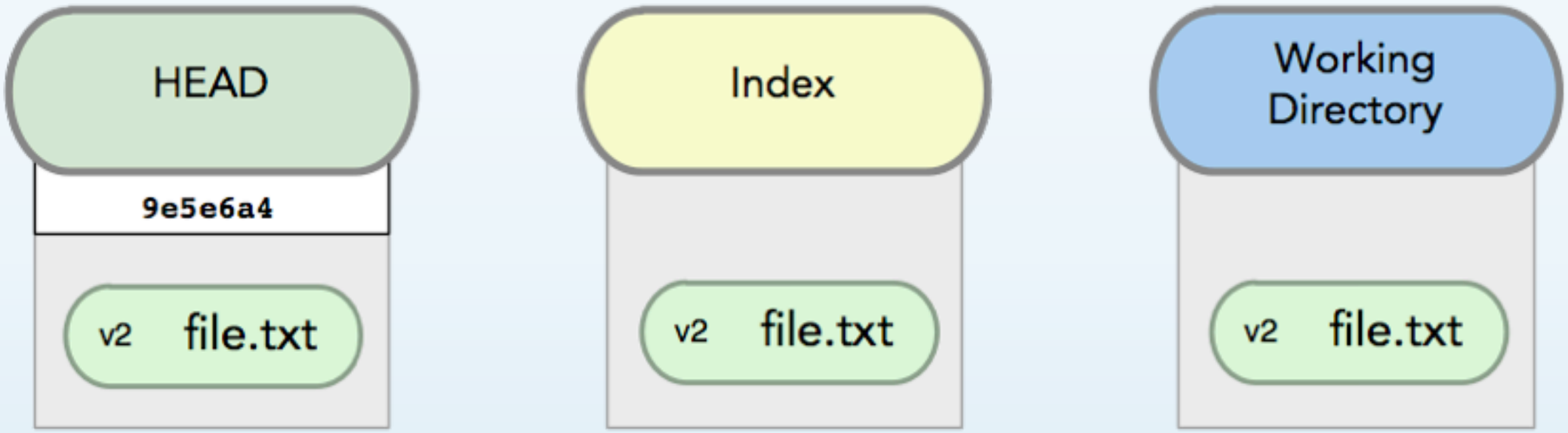
git add


HEAD
master
eb43bf8
9e5e6a4
v1 file.txt
v2 file.txt
Git Repository

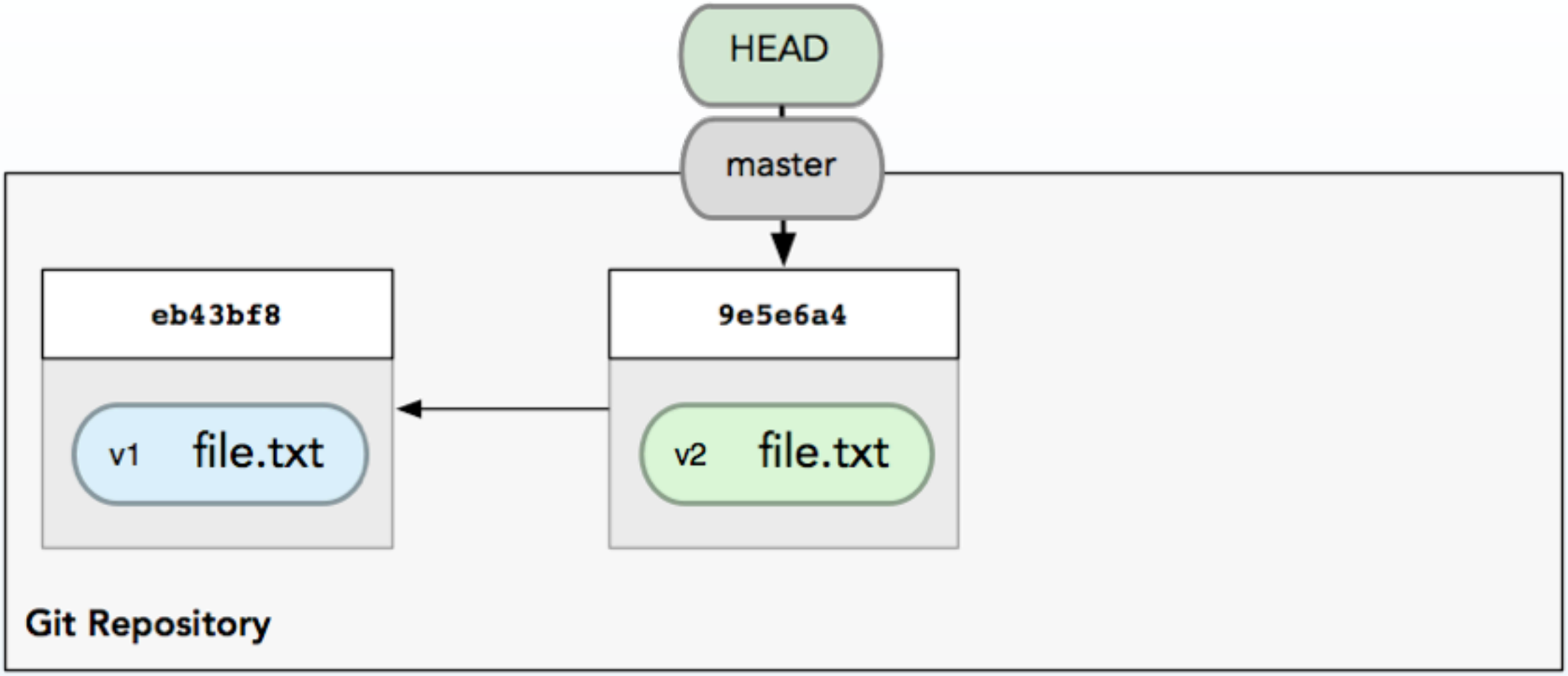

HEAD
9e5e6a4
v2 file.txt
Index
v2 file.txt
Working Directory
v2 file.txt

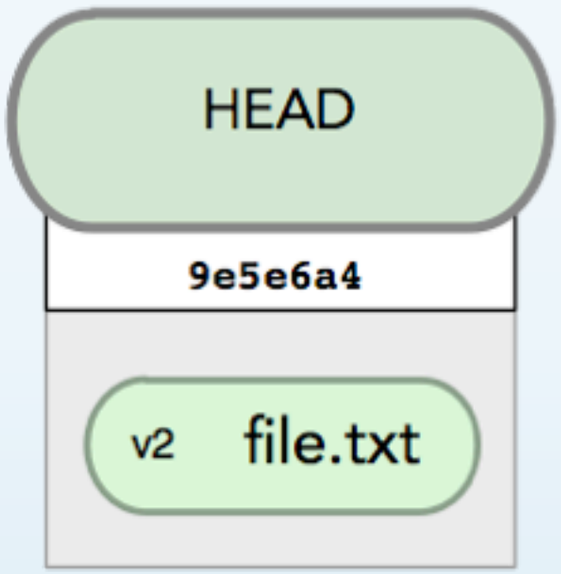
# git commit

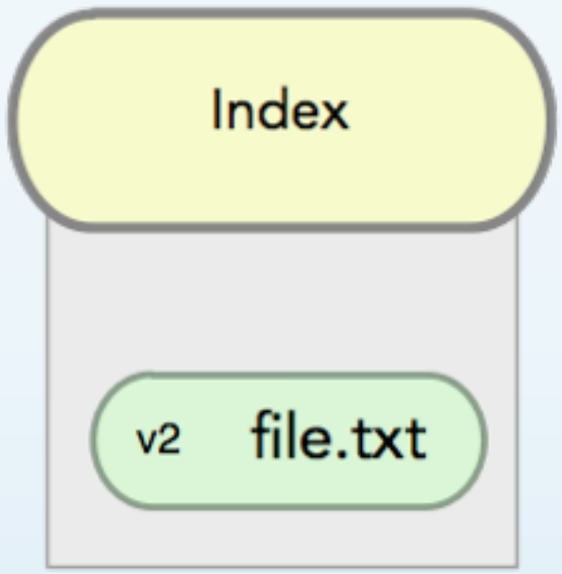

HEAD
master
eb43bf8
9e5e6a4
v1 file.txt
v2 file.txt
Git Repository

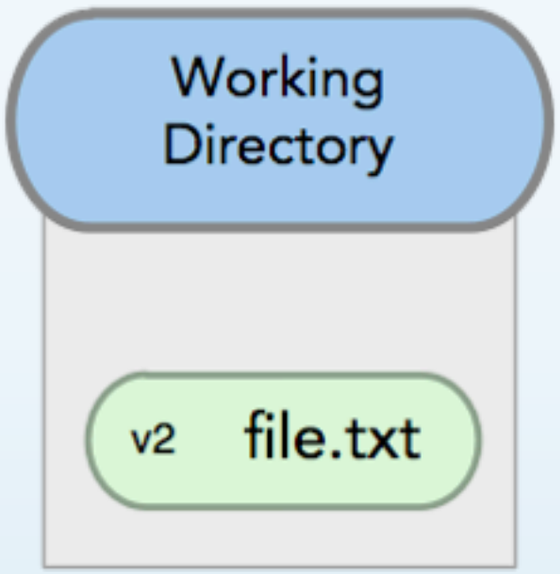

HEAD
9e5e6a4
v2 file.txt
Index
v2 file.txt
Working Directory
v2 file.txt

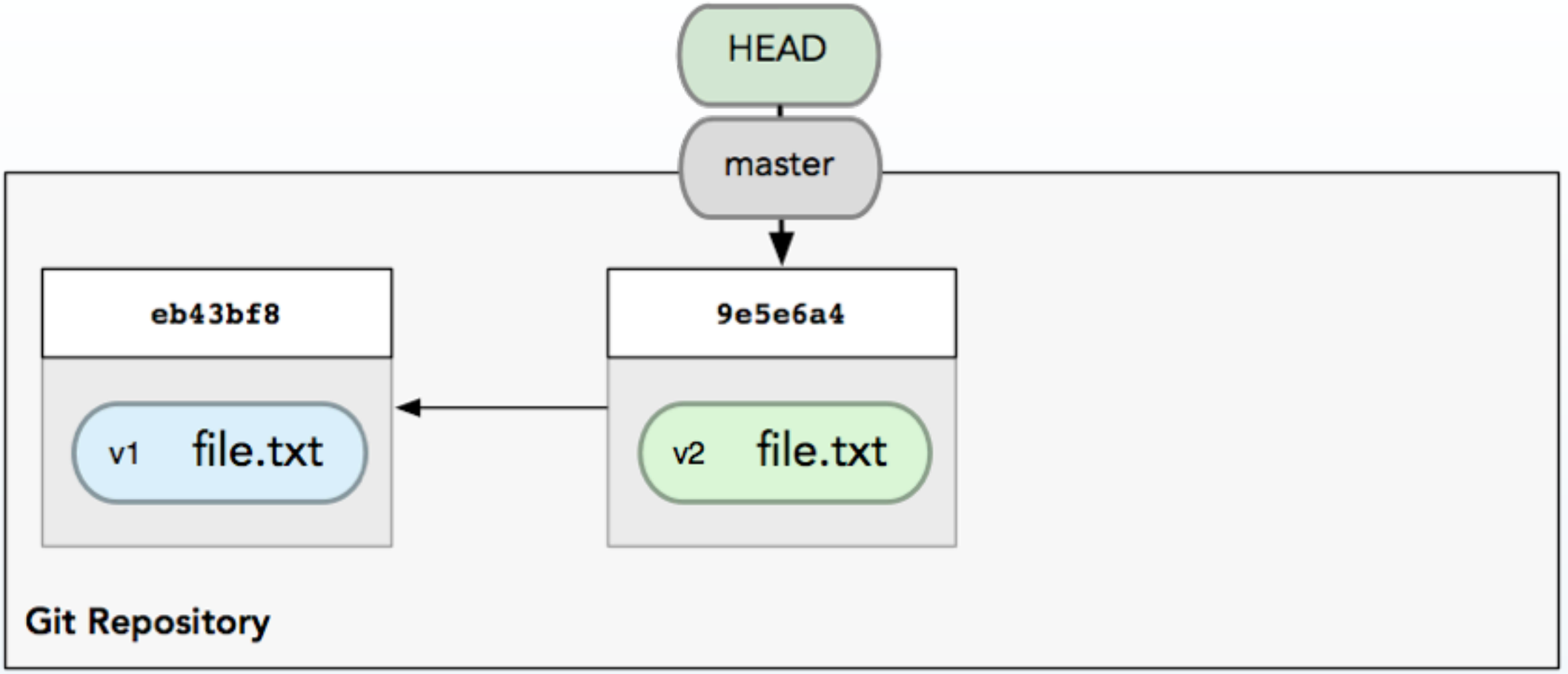
# git commit

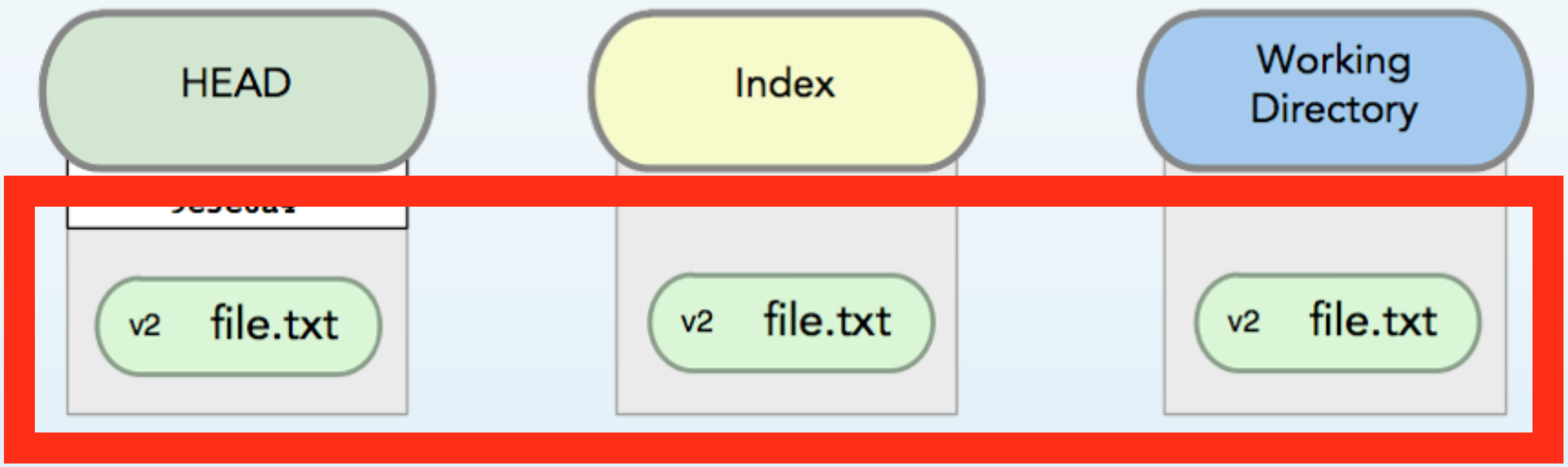

HEAD
master
eb43bf8
9e5e6a4
v1 file.txt
v2 file.txt
Git Repository


HEAD
9e5e6a4
v2 file.txt


Index
v2 file.txt


Working Directory
v2 file.txt


HEAD
master
eb43bf8
9e5e6a4
v1 file.txt
v2 file.txt
Git Repository


HEAD
Index
Working Directory
v2 file.txt
v2 file.txt
v2 file.txt

# git reset

# 2 tipos

```
git reset [commit] [ruta]
```

```
git reset [commit]
```

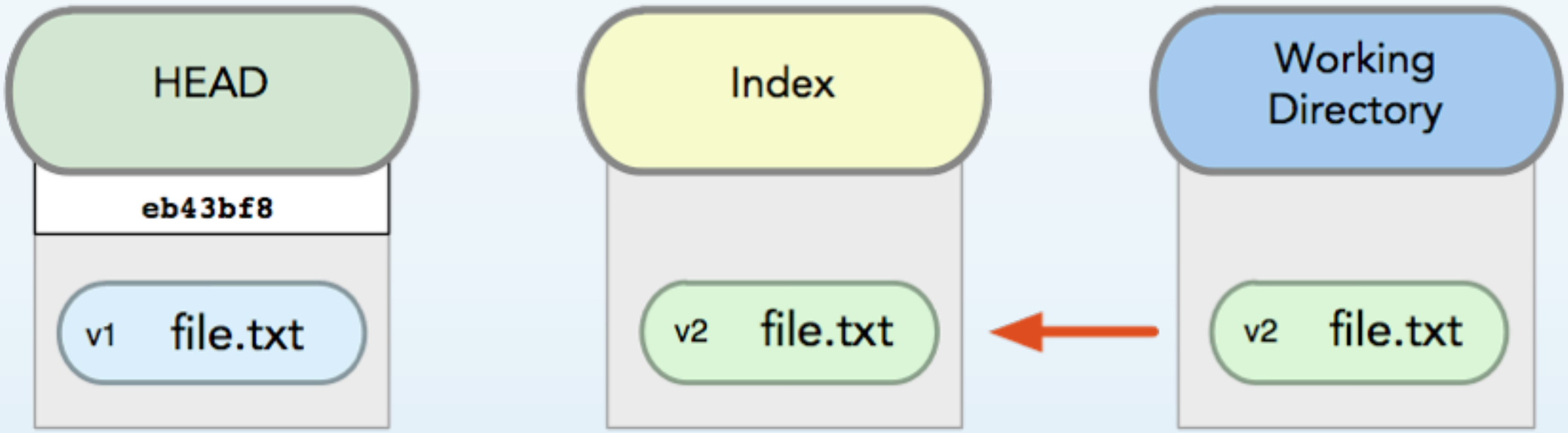
# 1. Forma Con Ruta

```
git reset [commit] [ruta]
```

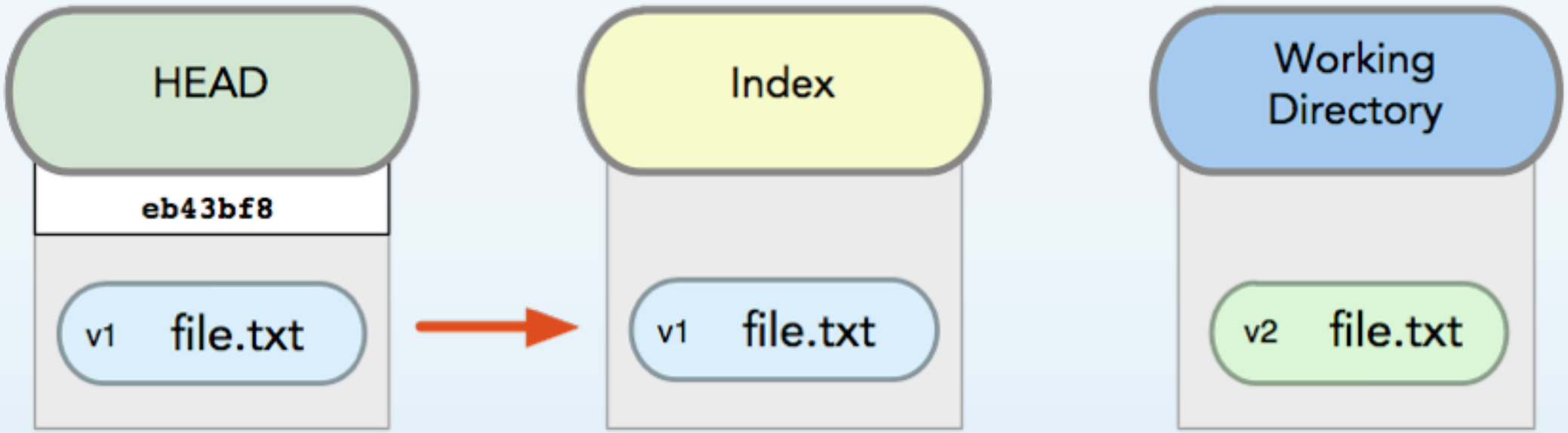
`git reset [file]`

**es el opuesto de**

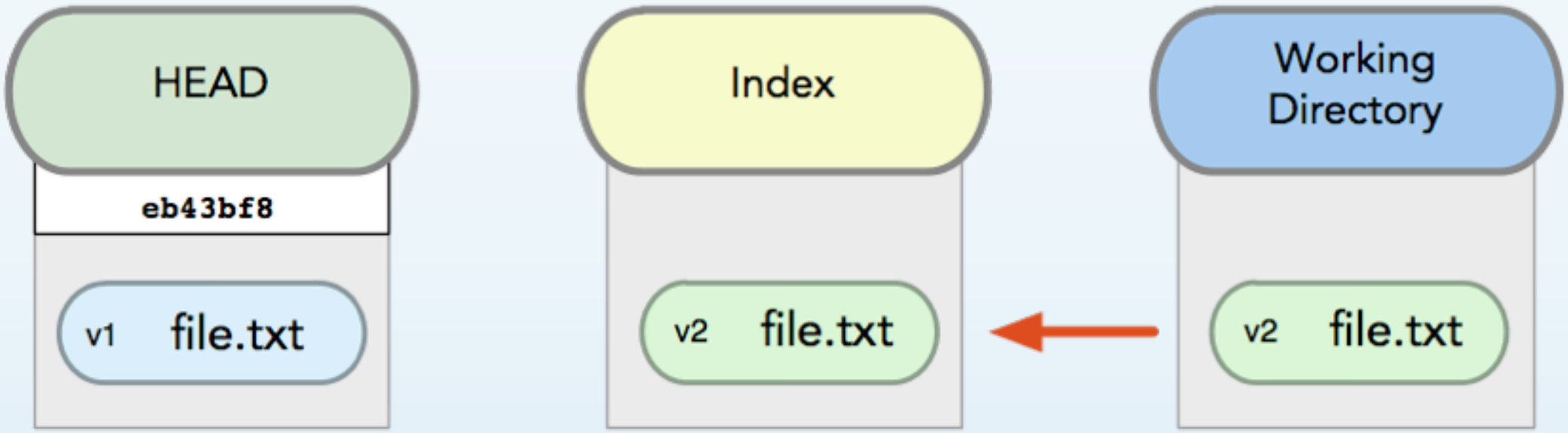
`git add [file]`

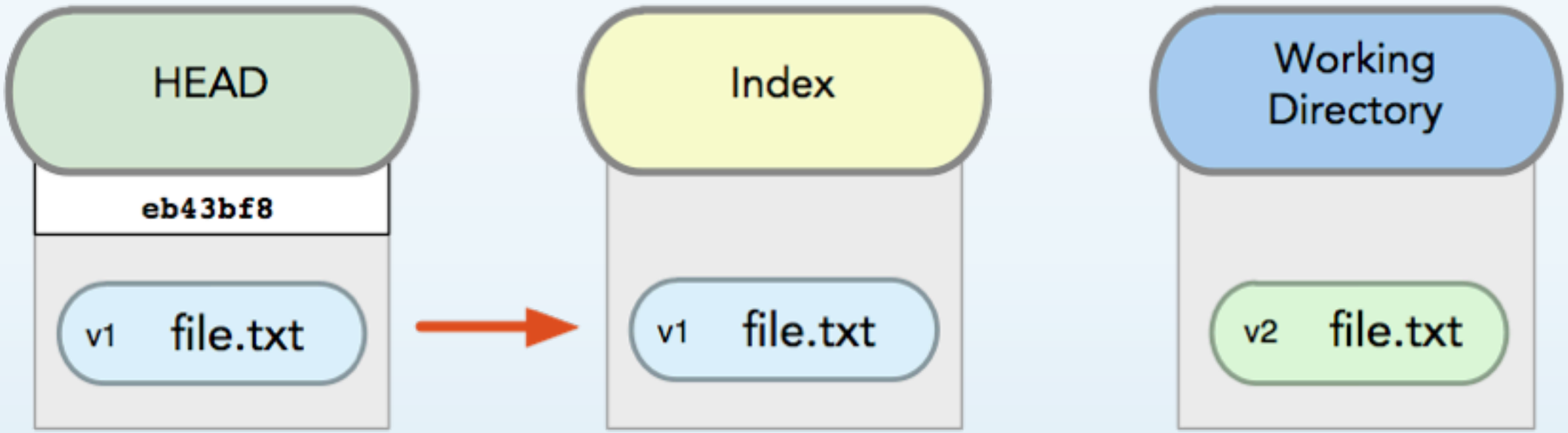

# git add file.txt


# git reset file.txt


# git add file.txt


# git reset file.txt

# 2. Forma Sin Ruta

```
git reset [commit]
```

# Las Opciones de Reset

# Las Opciones de Reset

**--soft** mueve HEAD al objetivo

# Las Opciones de Reset

**--soft** mueve HEAD al objetivo

**[--mixed]** copia al índice

# Las Opciones de Reset

**--soft** mueve HEAD al objetivo

**[--mixed]** copia al índice

**--hard** copia al dir de trabajo

# --hard

# __

# mal

reset

!=

mal

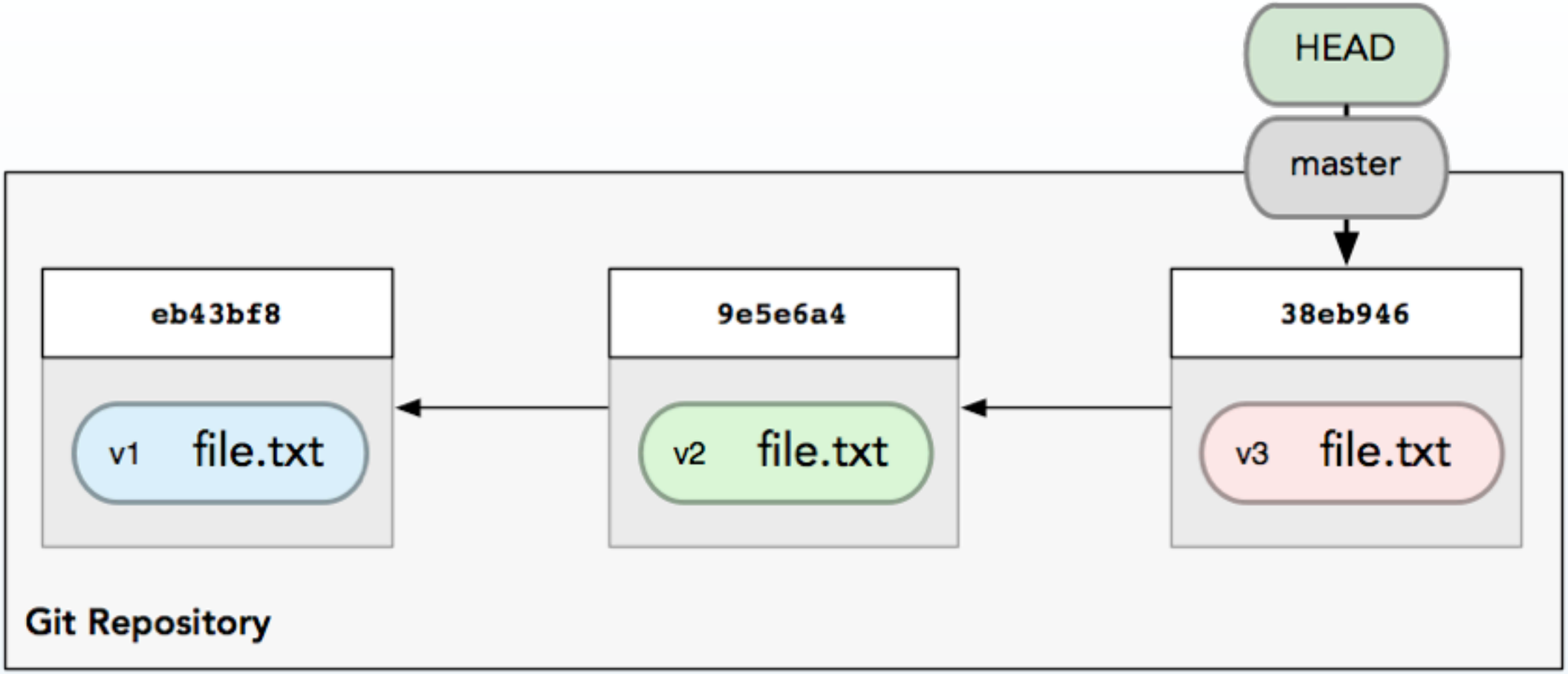
a excepción de **--hard**

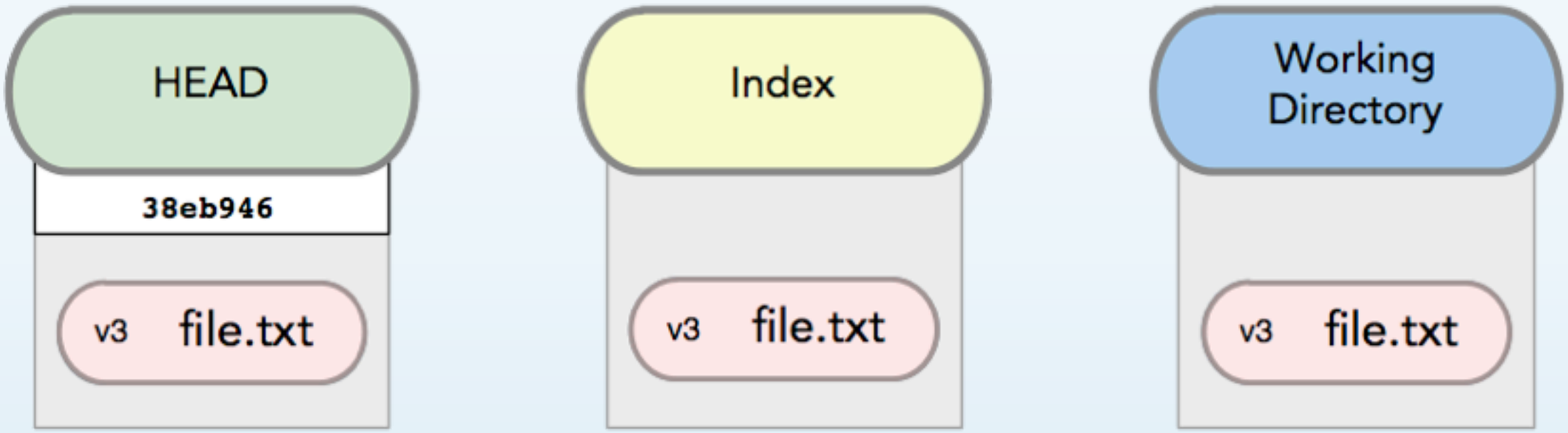
# reset != mal

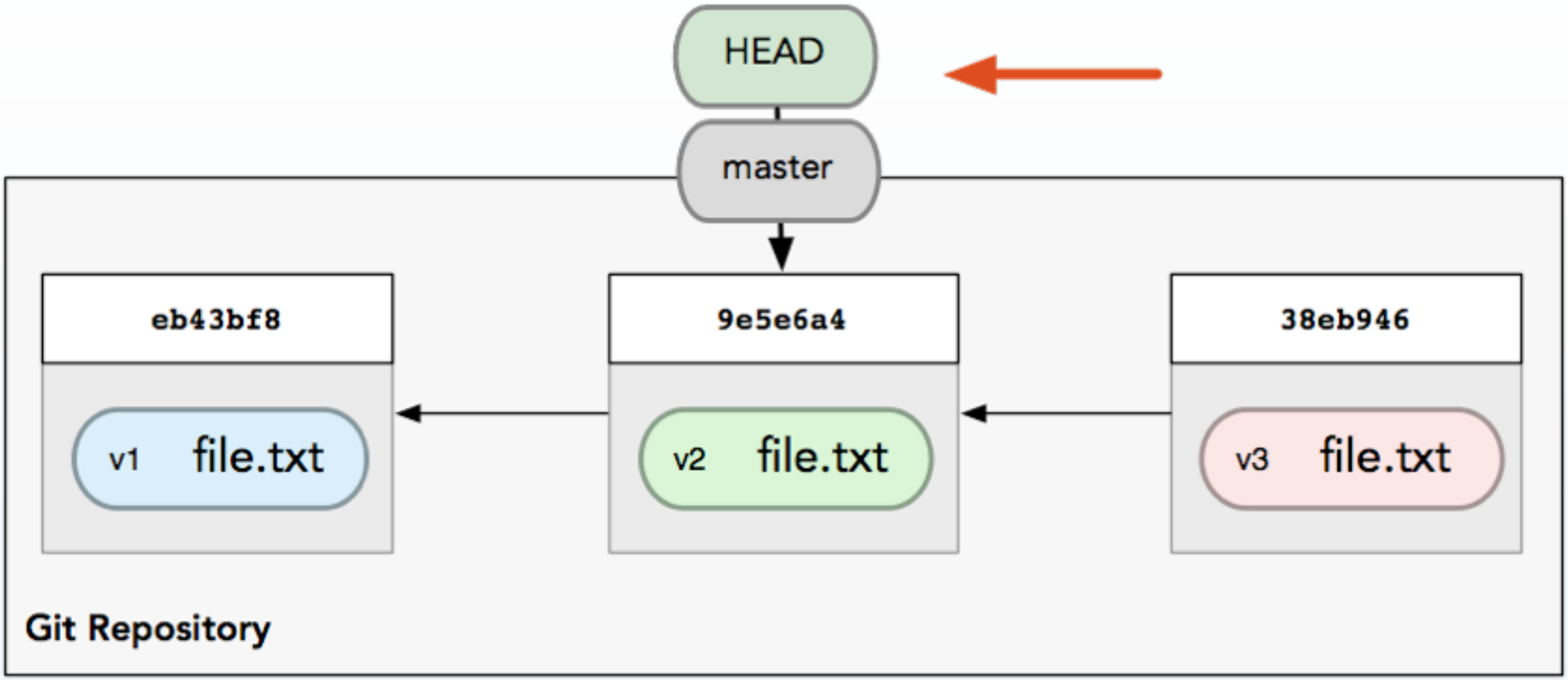
# revisemos

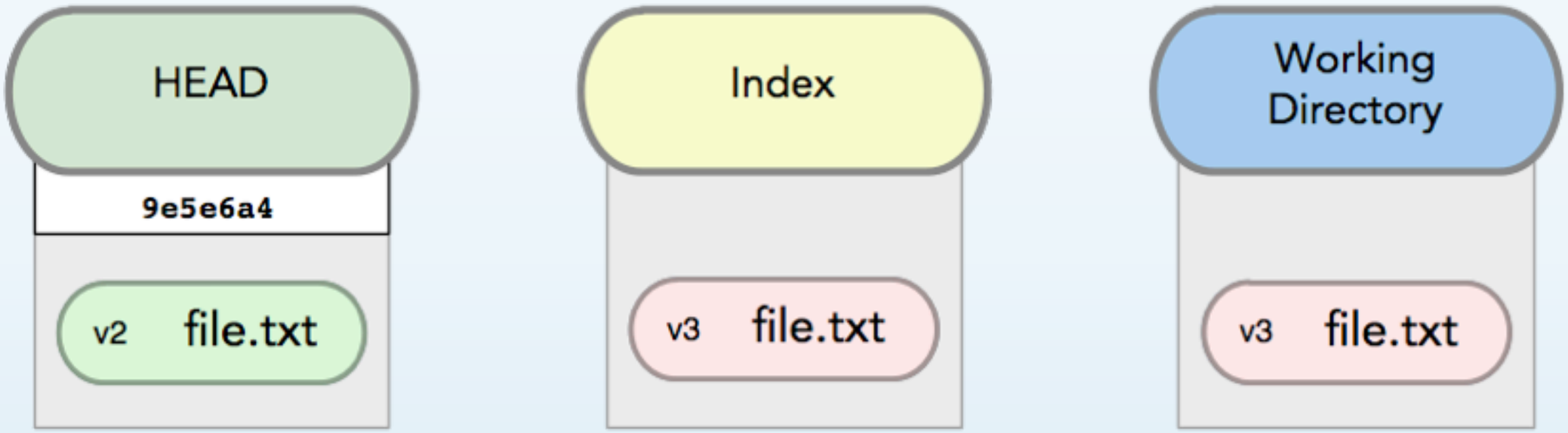
# --soft

mueve HEAD a otro commit

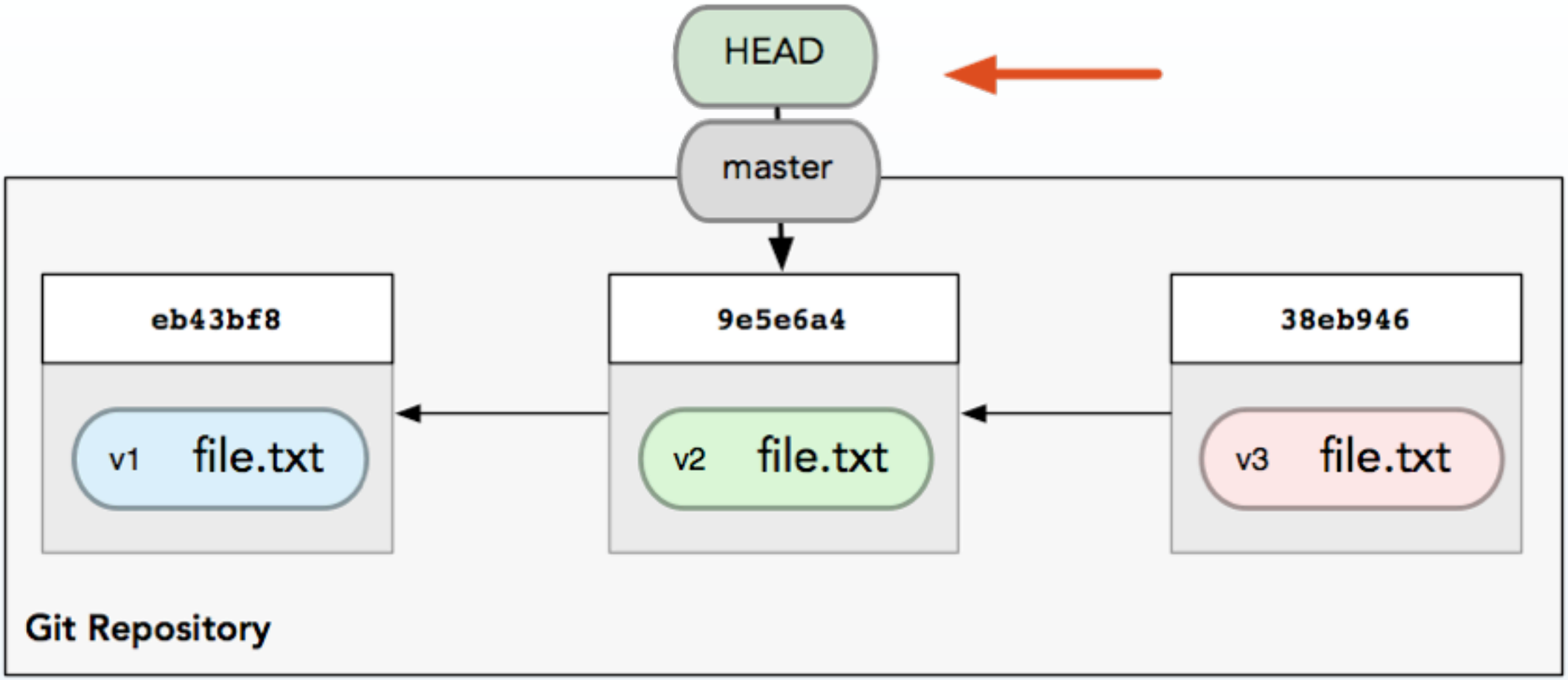

HEAD
master
eb43bf8
9e5e6a4
38eb946
v1 file.txt
v2 file.txt
v3 file.txt
Git Repository

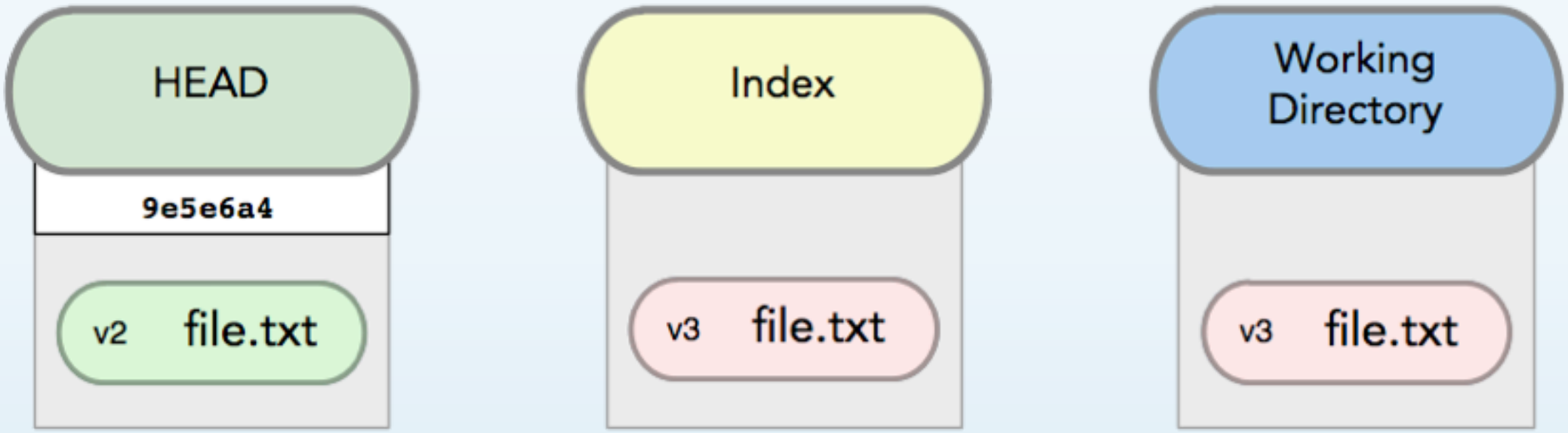

HEAD
38eb946
v3 file.txt
Index
v3 file.txt
Working Directory
v3 file.txt



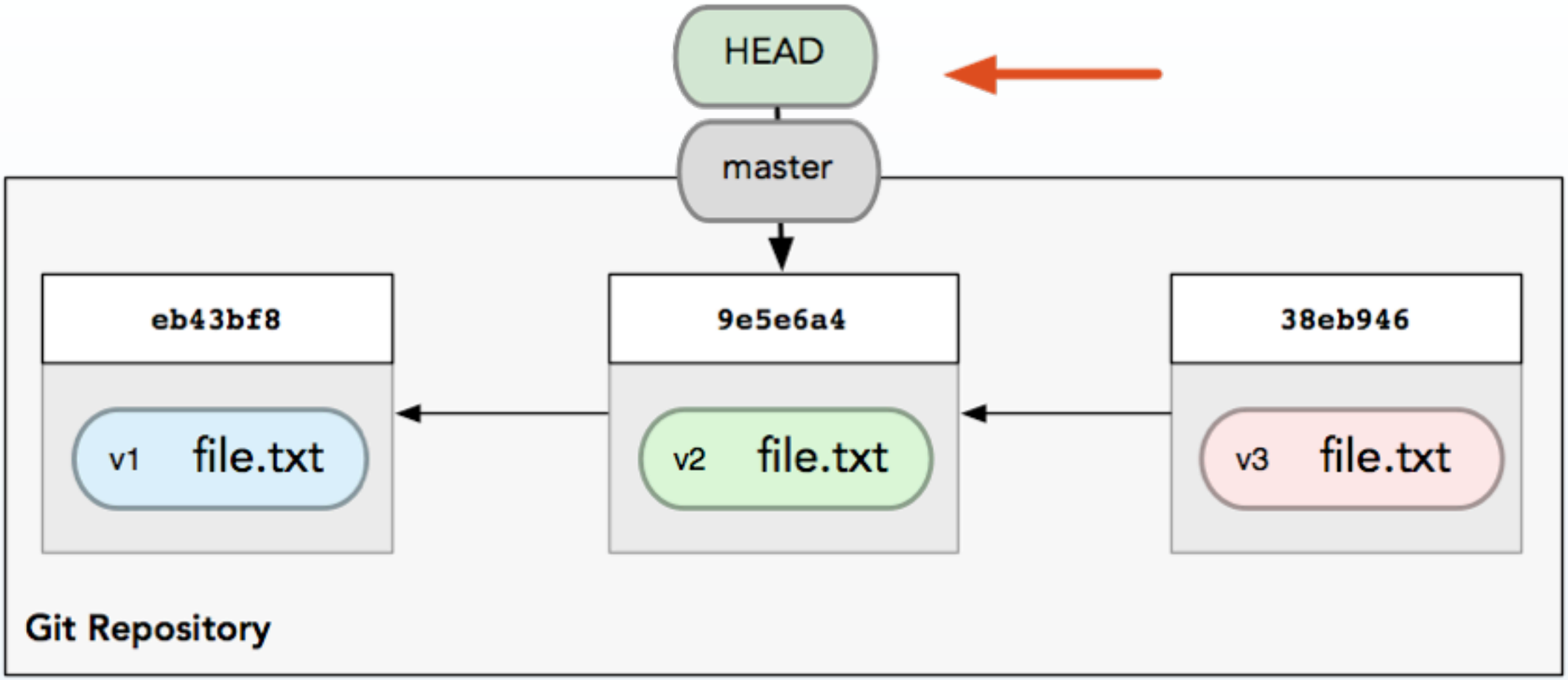



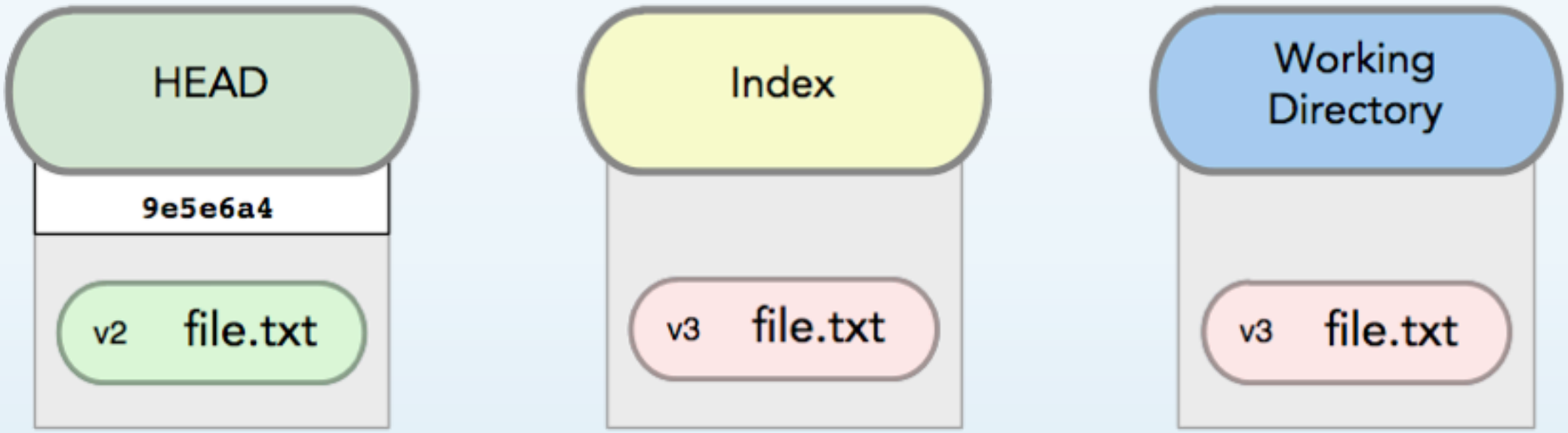

# git reset --soft HEAD~




# git reset --soft HEAD~




# git reset --soft HEAD~

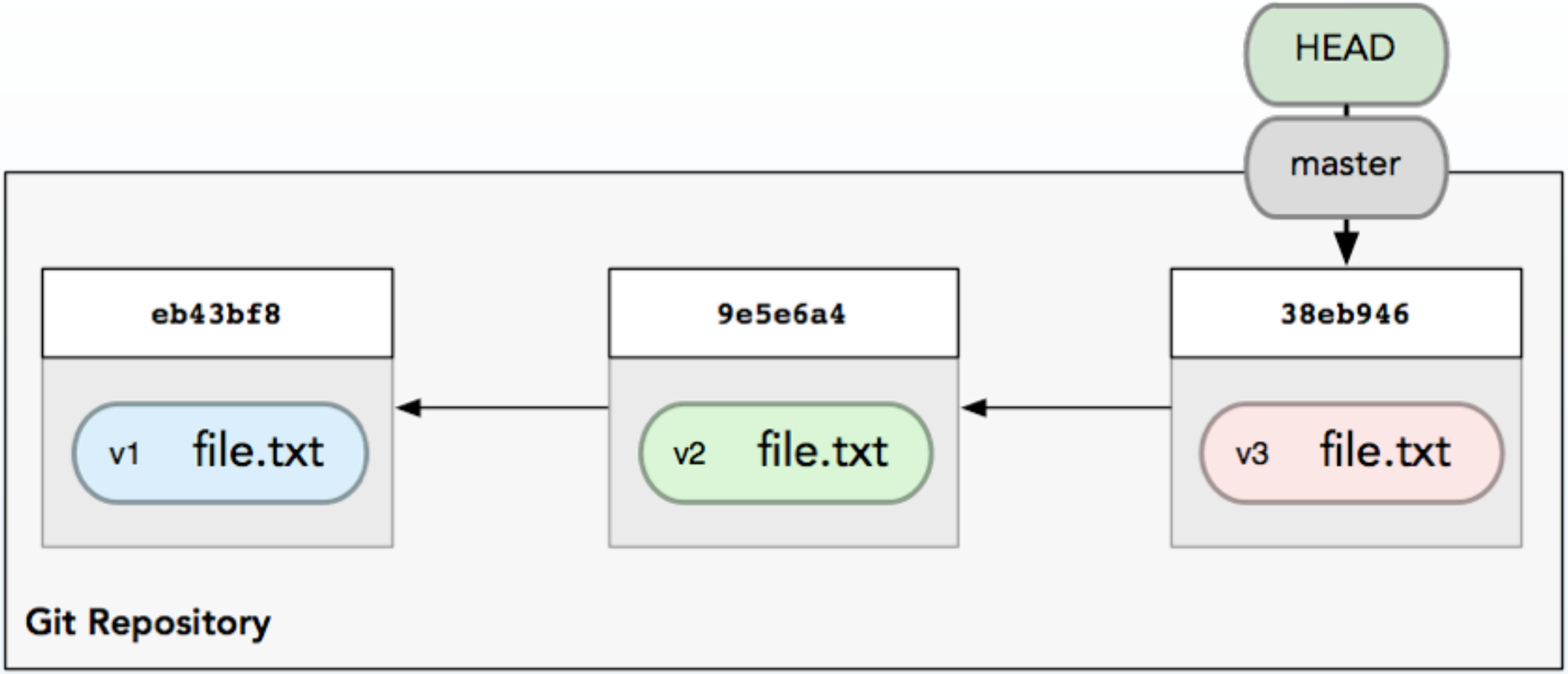
# git reset --soft HEAD~

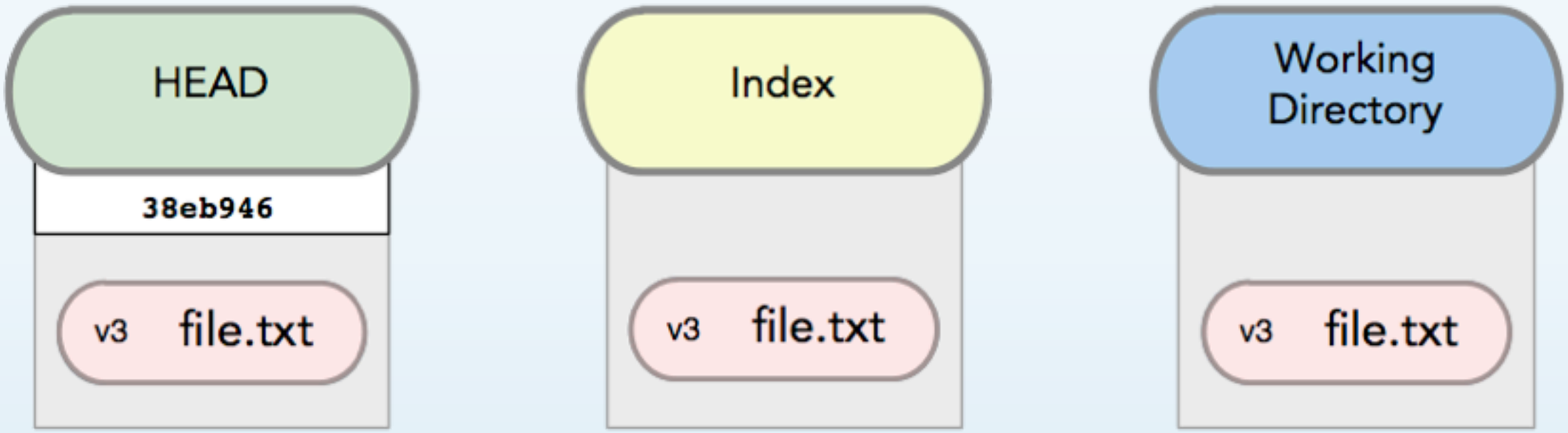
es como que

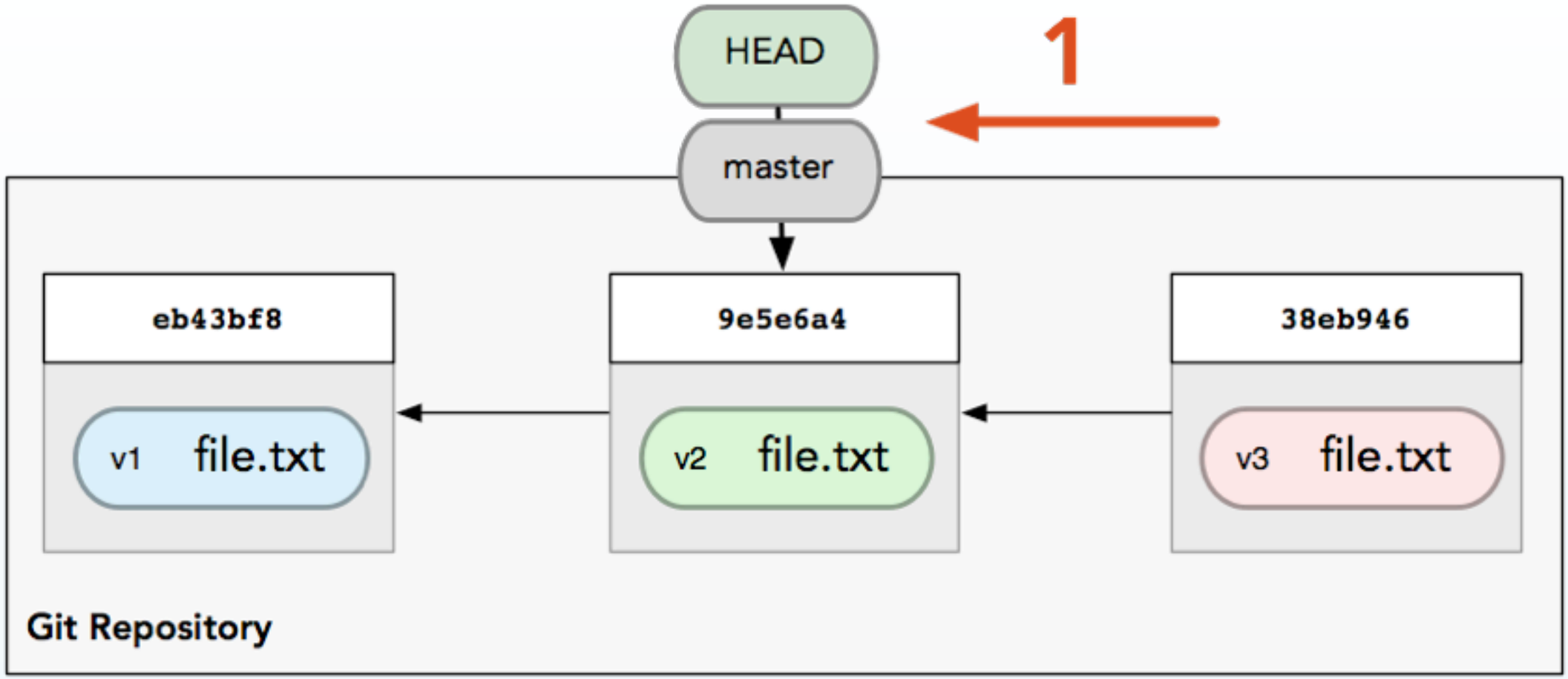
# git uncommit

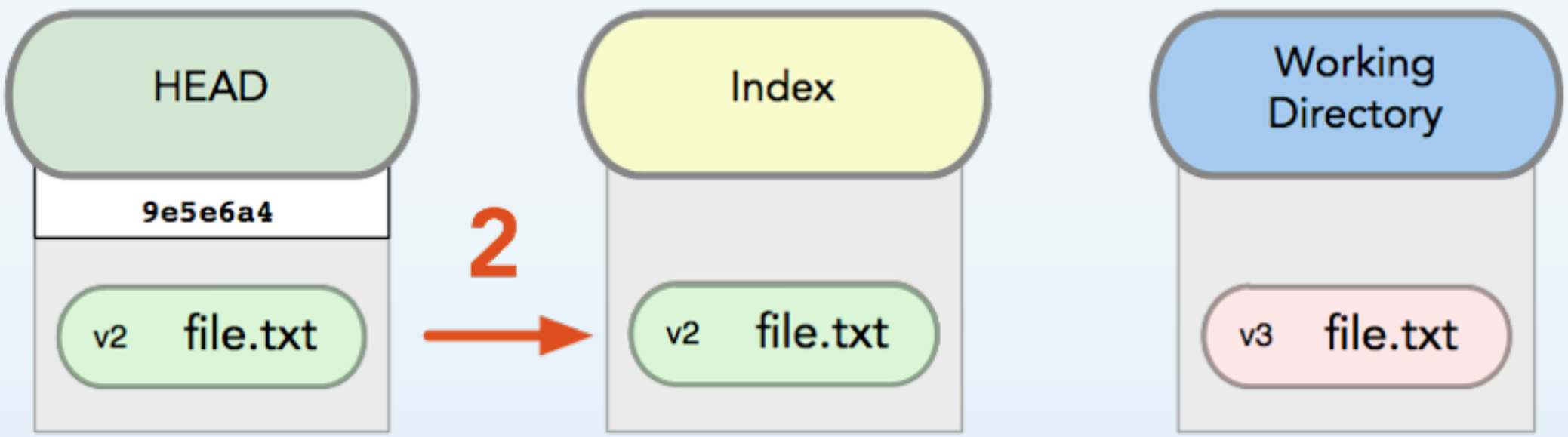
# --mixed

mueve HEAD a otro commit, y despues lo copia al índice


HEAD
master
eb43bf8
9e5e6a4
38eb946
v1 file.txt
v2 file.txt
v3 file.txt
Git Repository


HEAD
38eb946
v3 file.txt
Index
v3 file.txt
Working Directory
v3 file.txt




# git reset [--mixed] HEAD~

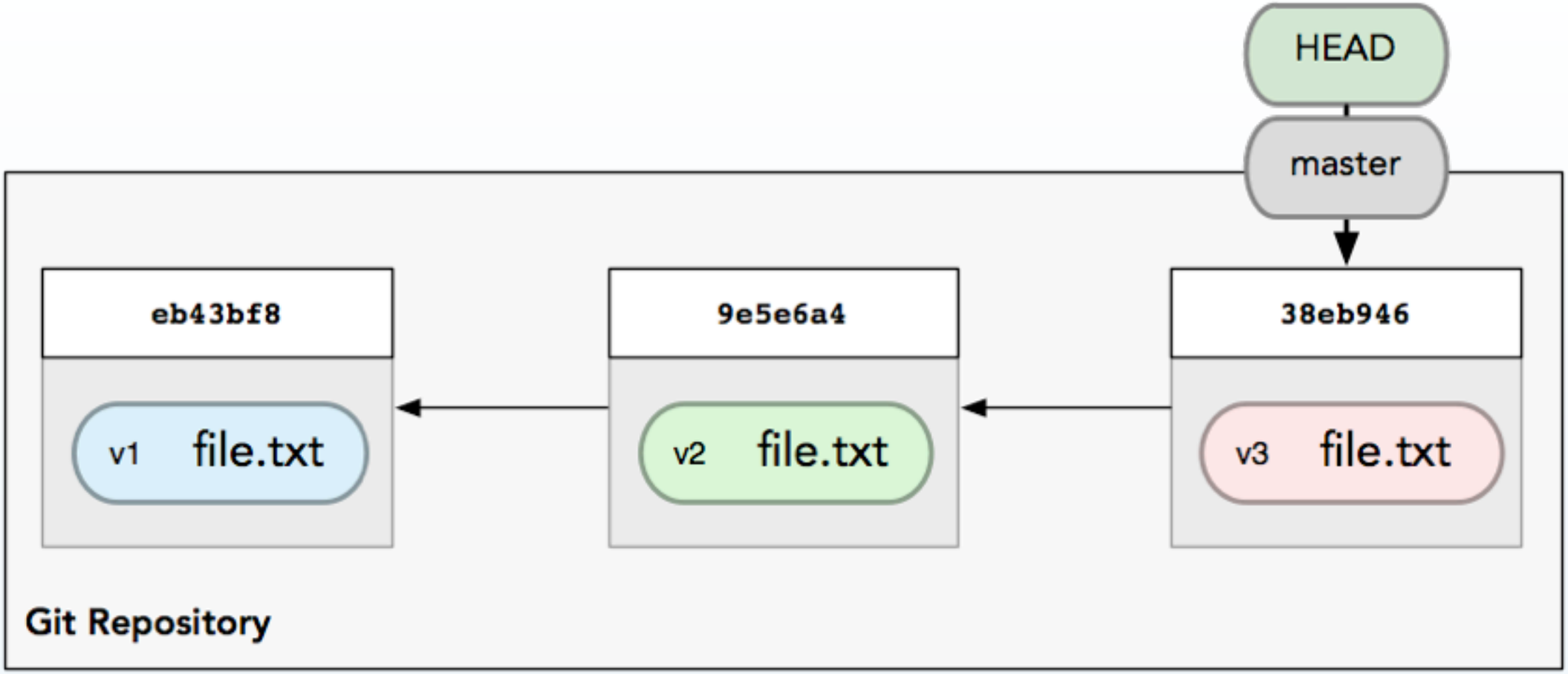
# git reset HEAD~

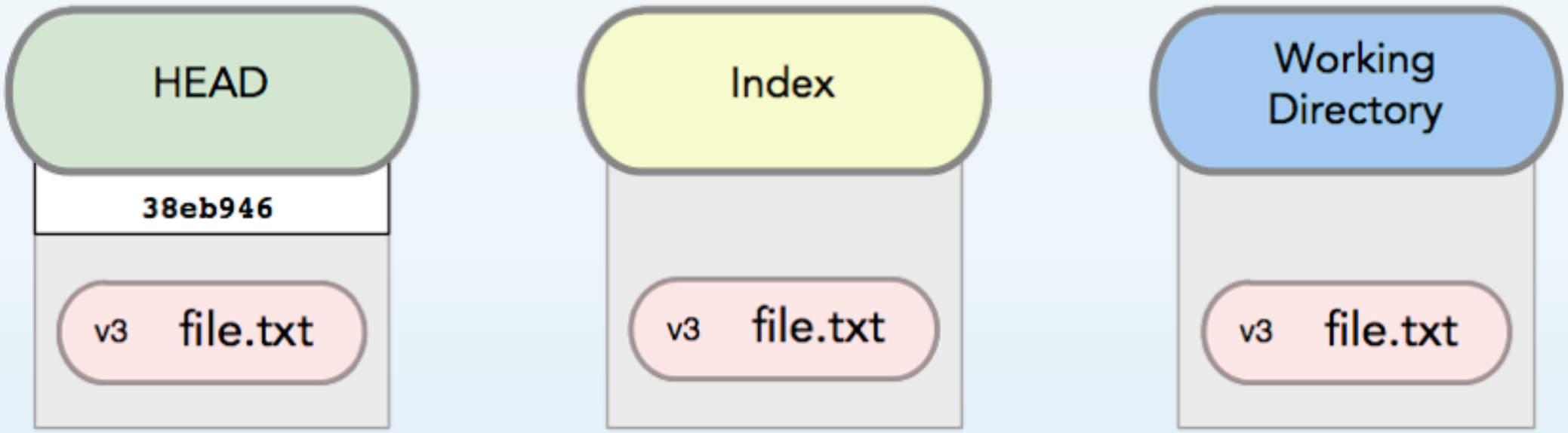
es como que

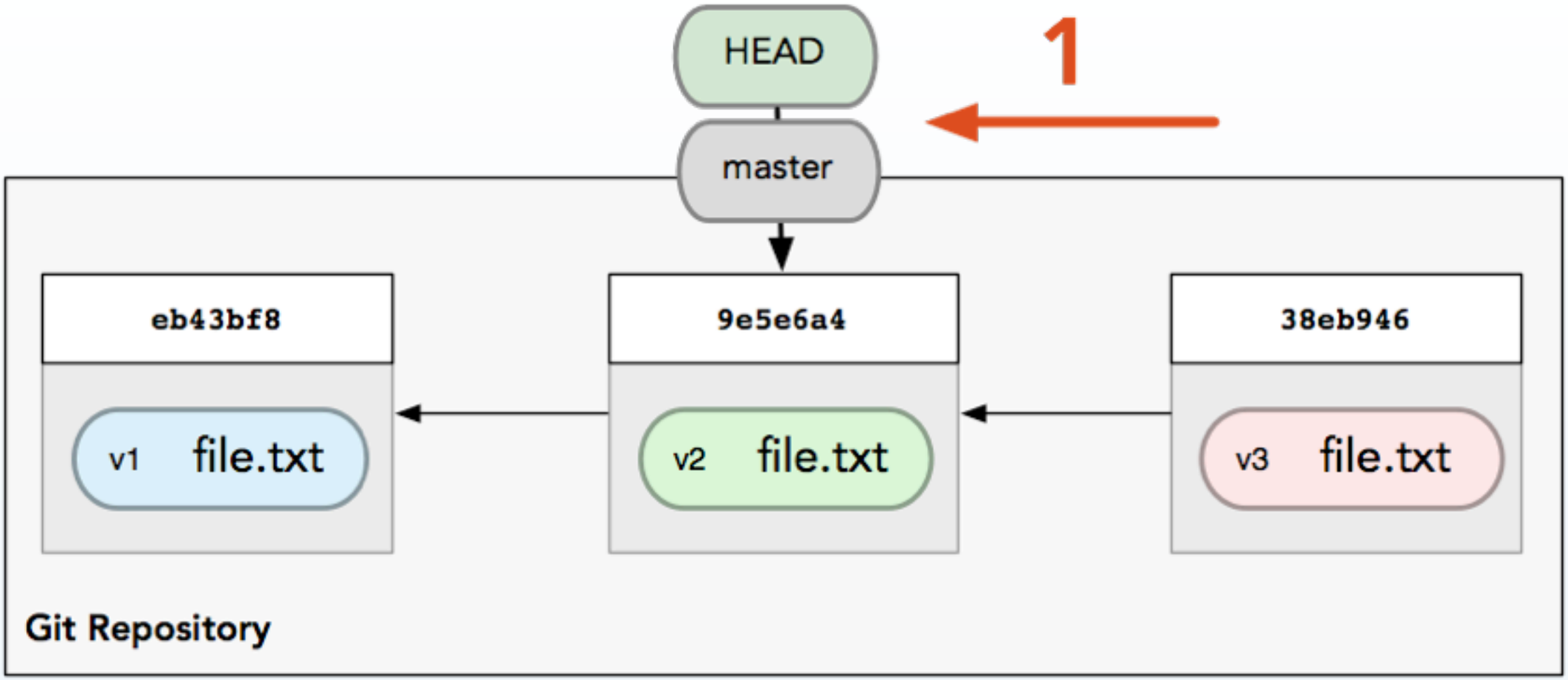
# git uncommit && git unadd *

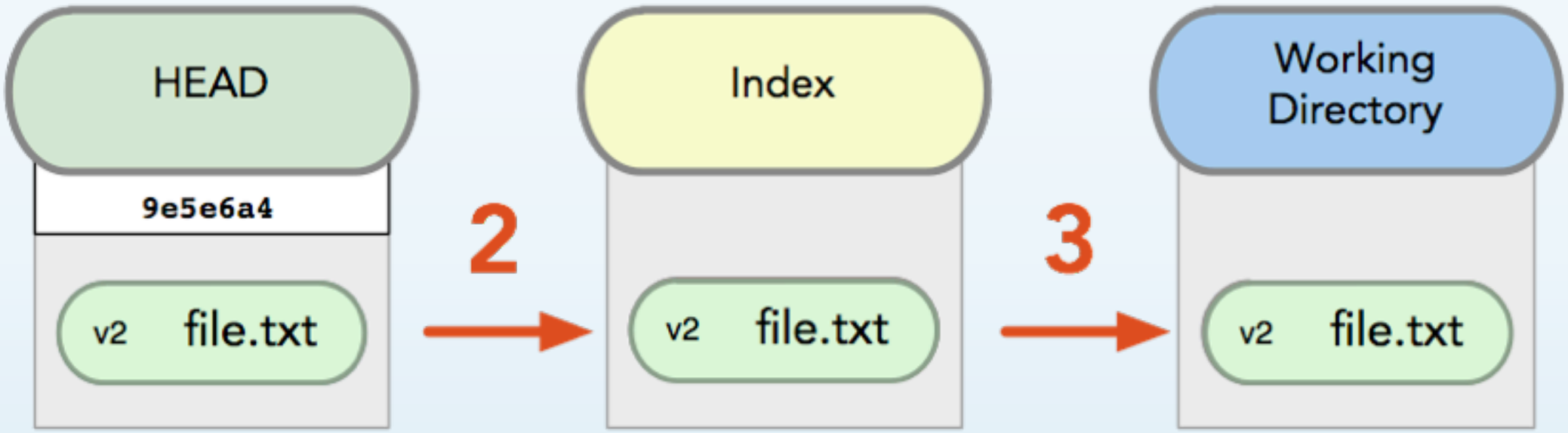
# --hard

mueve HEAD, copia al índice, y luego lo copia al directorio de trabajo


HEAD
master
eb43bf8
9e5e6a4
38eb946
v1 file.txt
v2 file.txt
v3 file.txt
Git Repository


HEAD
38eb946
v3 file.txt
Index
v3 file.txt
Working Directory
v3 file.txt




# git reset --hard HEAD~

# git reset --hard HEAD~

es como que

**git uncommit**
**git unadd ***
**ctrl-z ctrl-z ctrl-z**

# ¿para que?

# ¿para que?

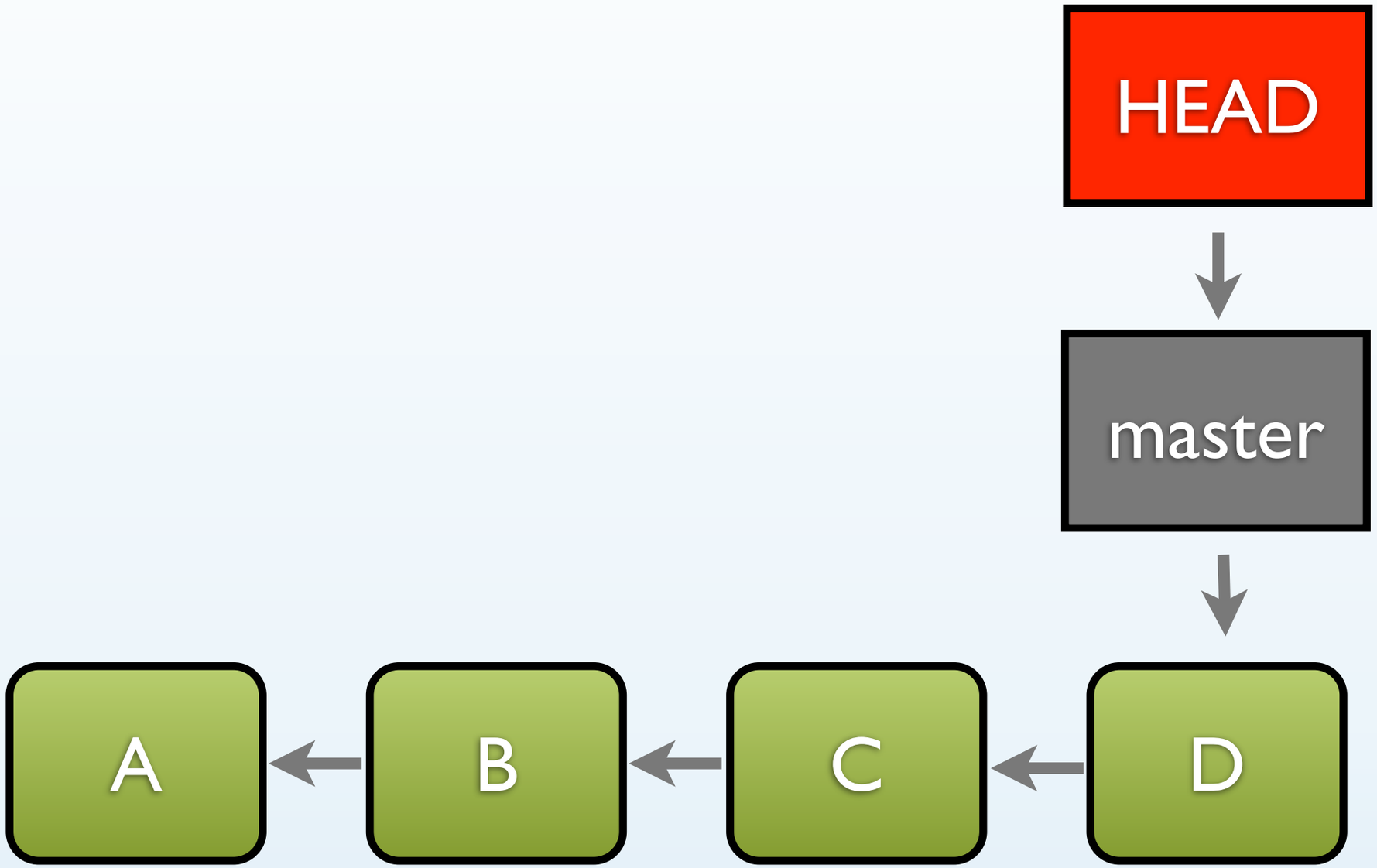
deshacer un commit

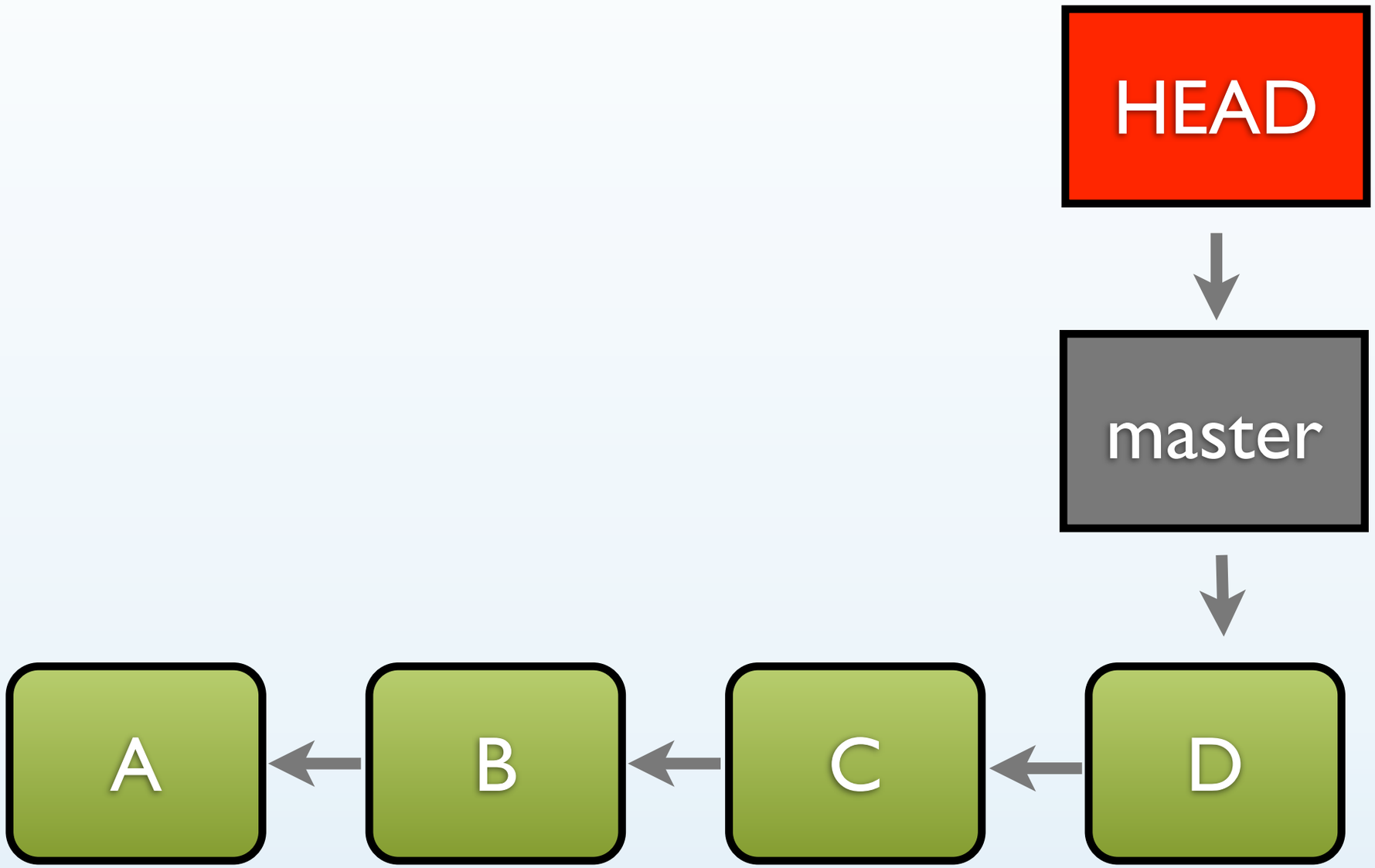
# ¿para que?

deshacer un commit

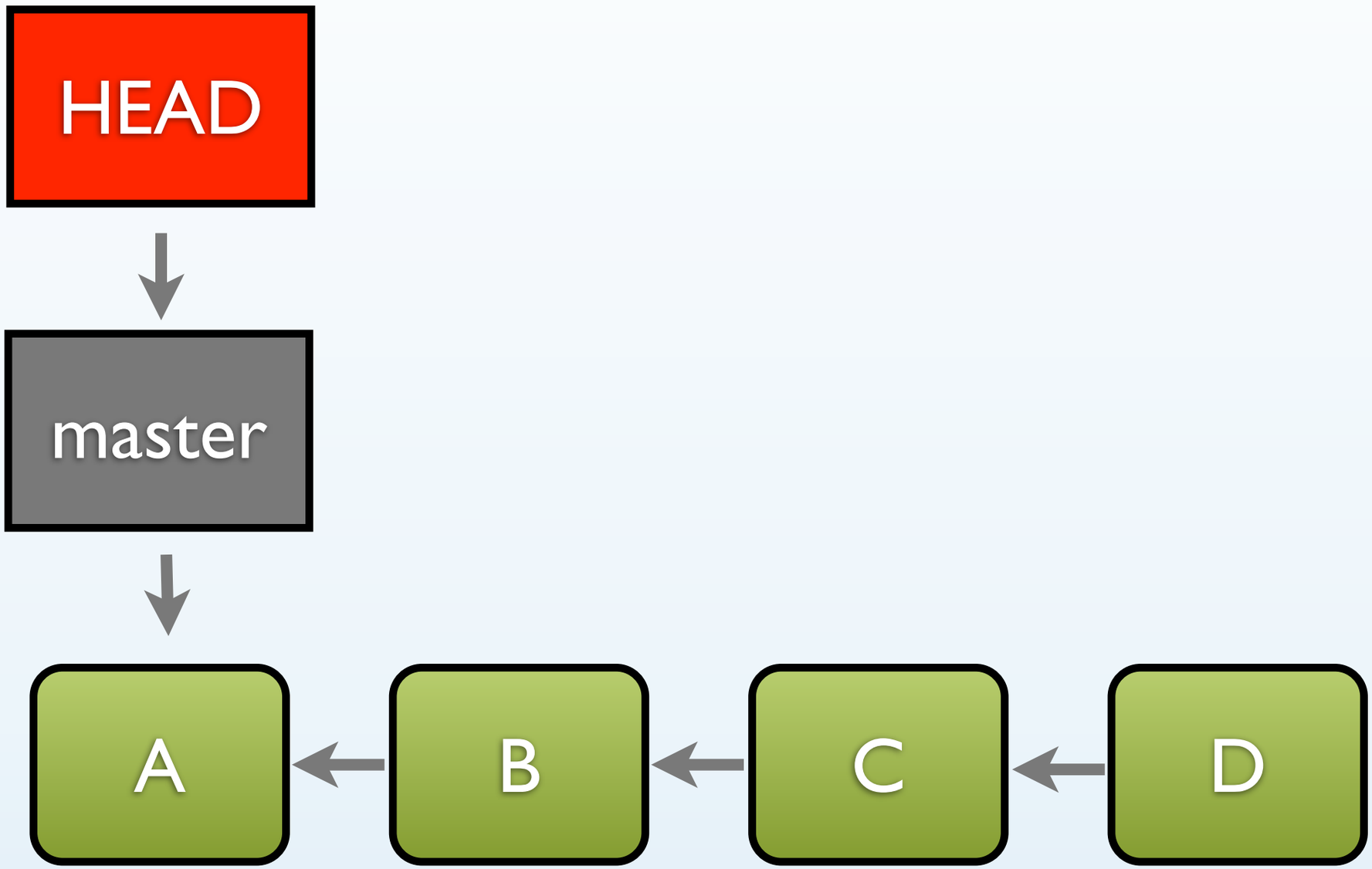
revertir fácilmente un archivo

# squashing

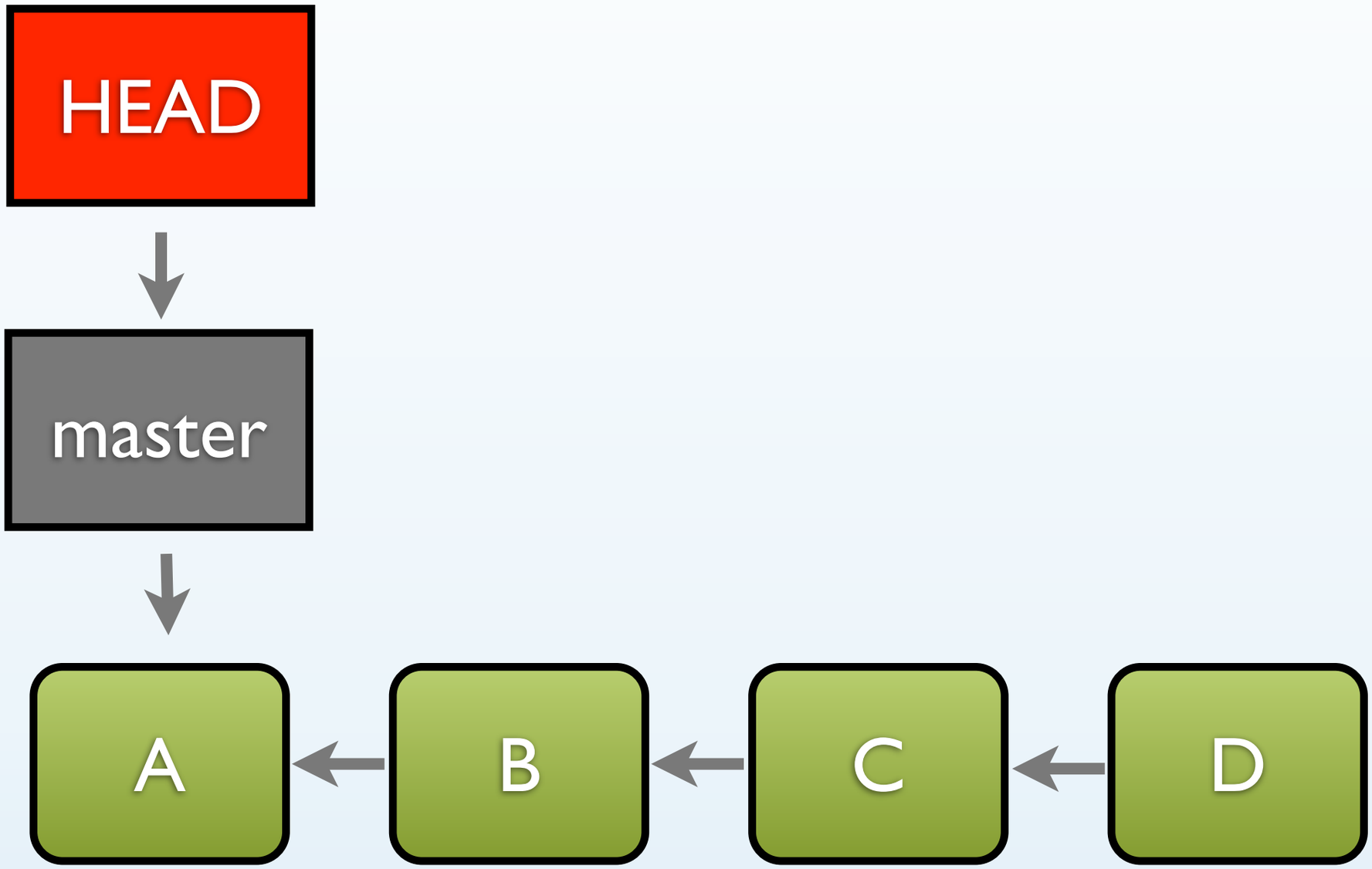

HEAD
master
A
B
C
D



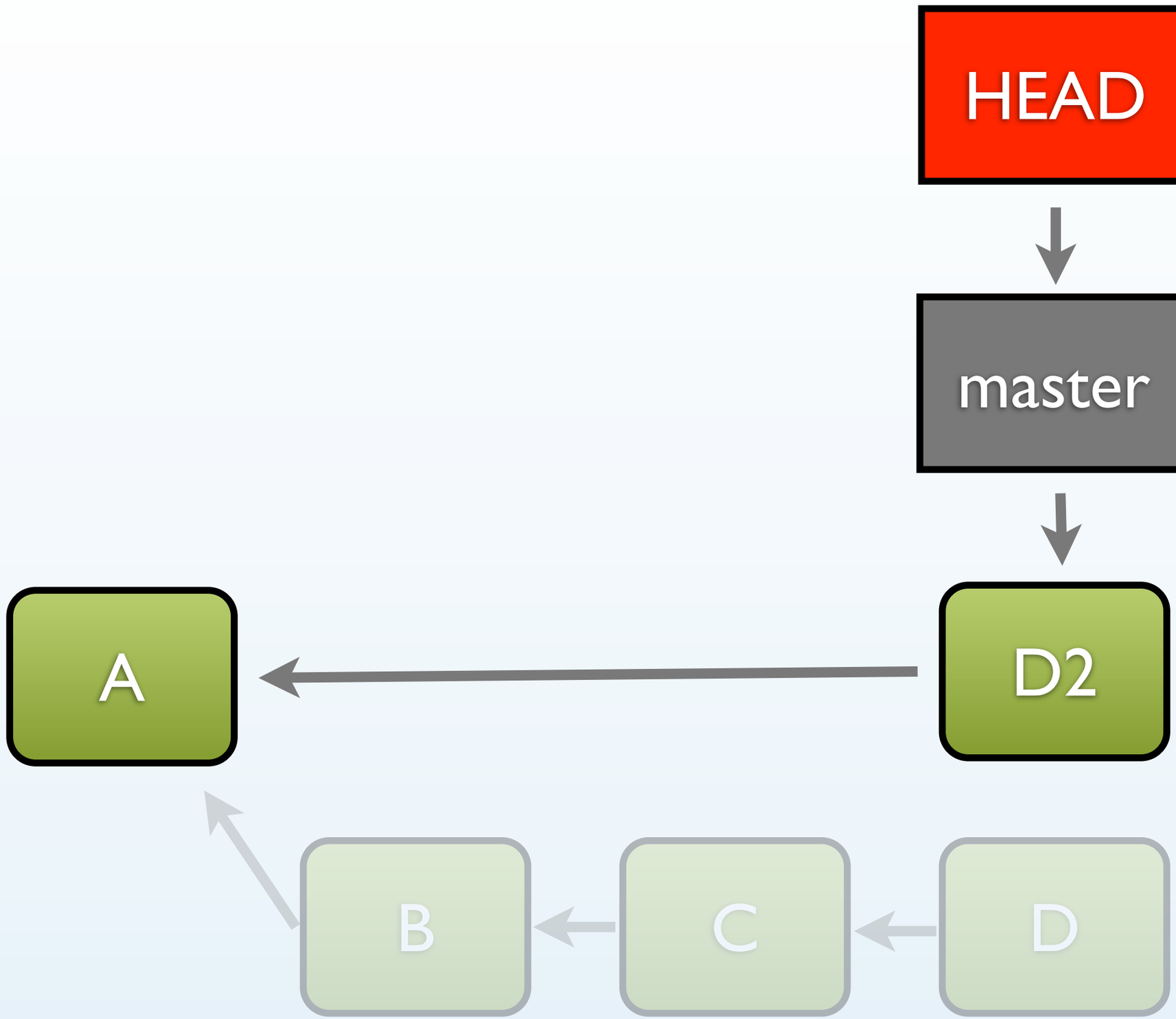

git reset --soft A



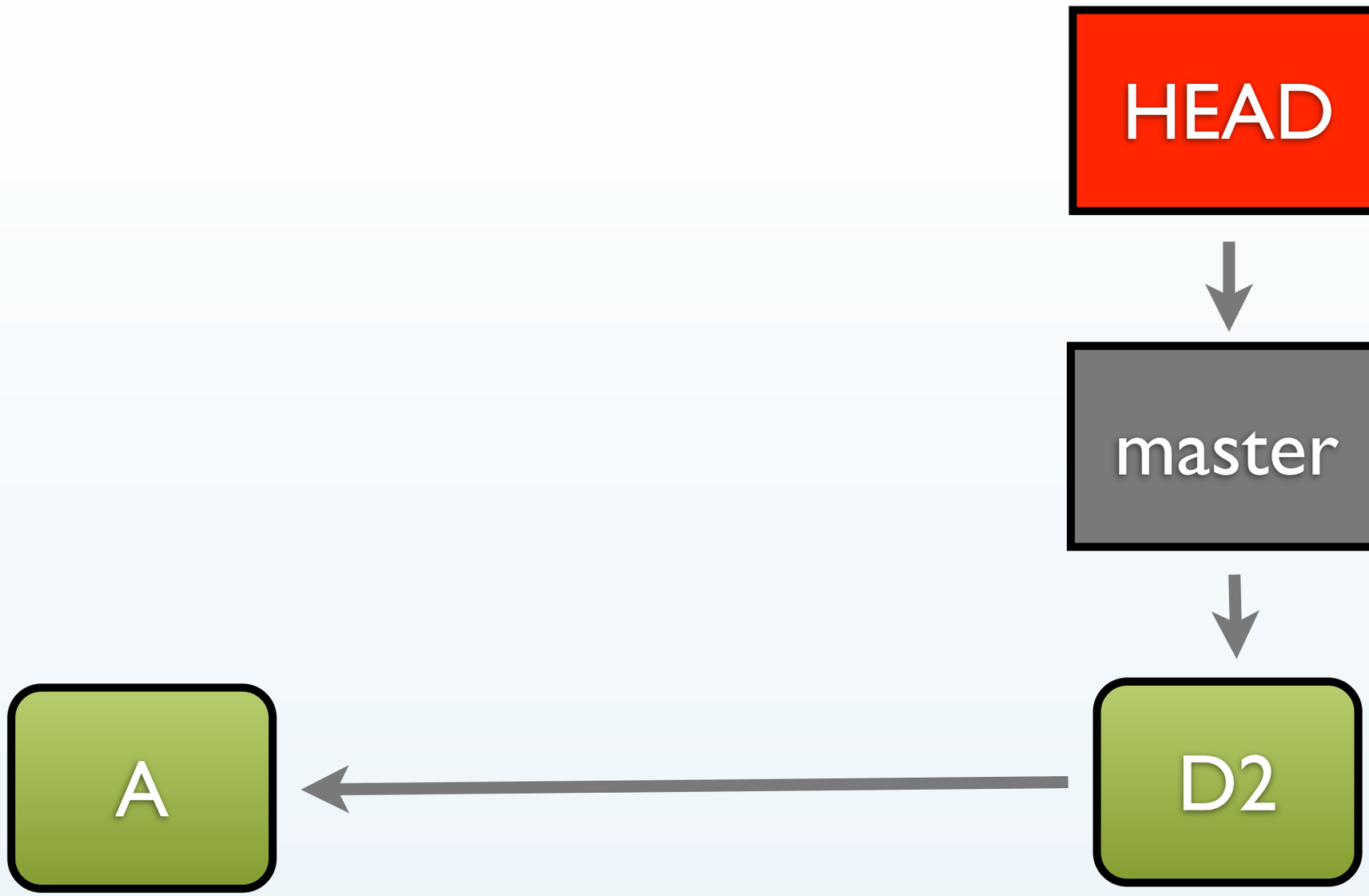

git reset --soft A

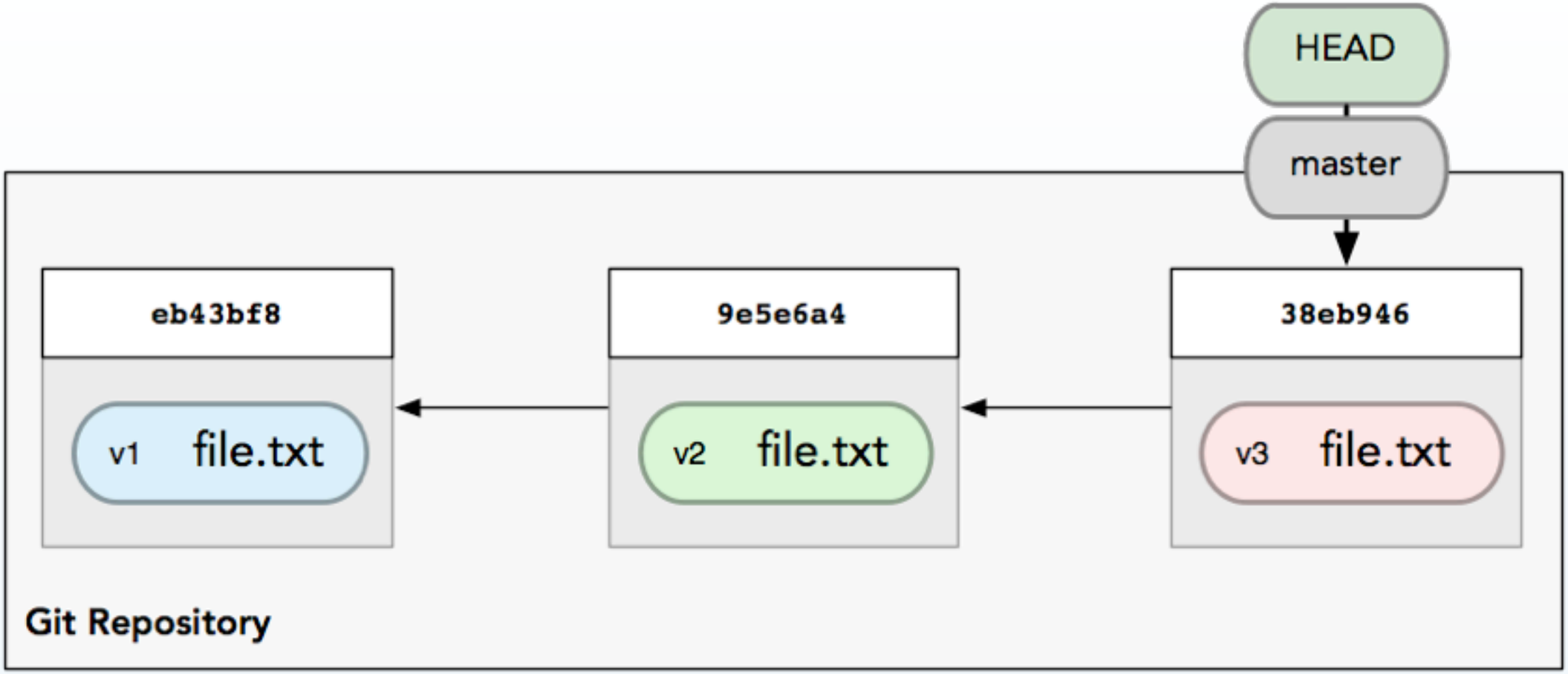


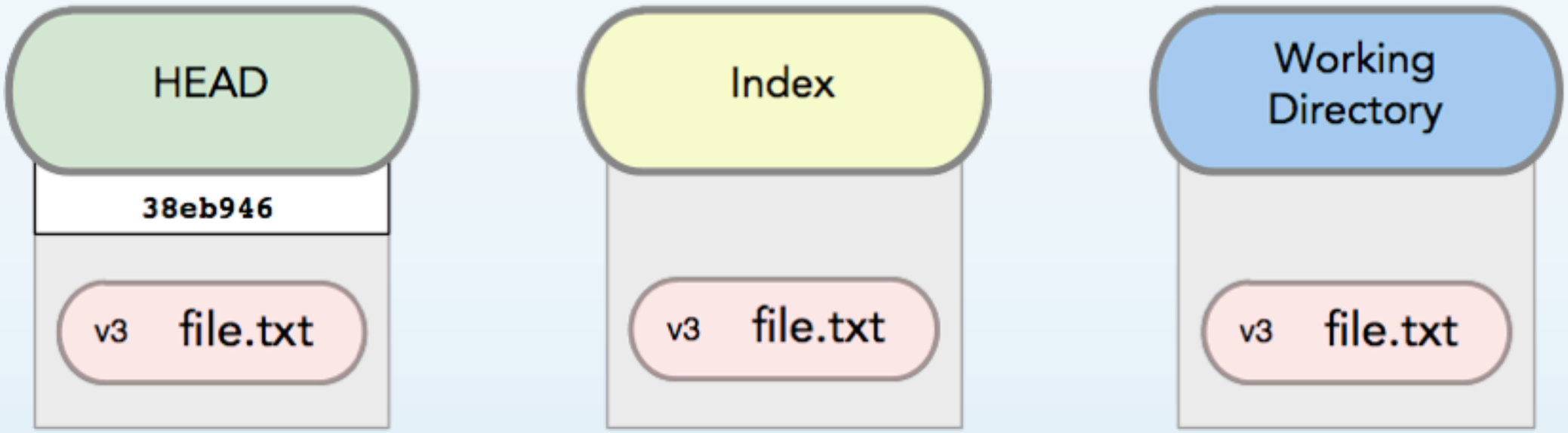

git commit -m ‘Yo soy muy inteligente’

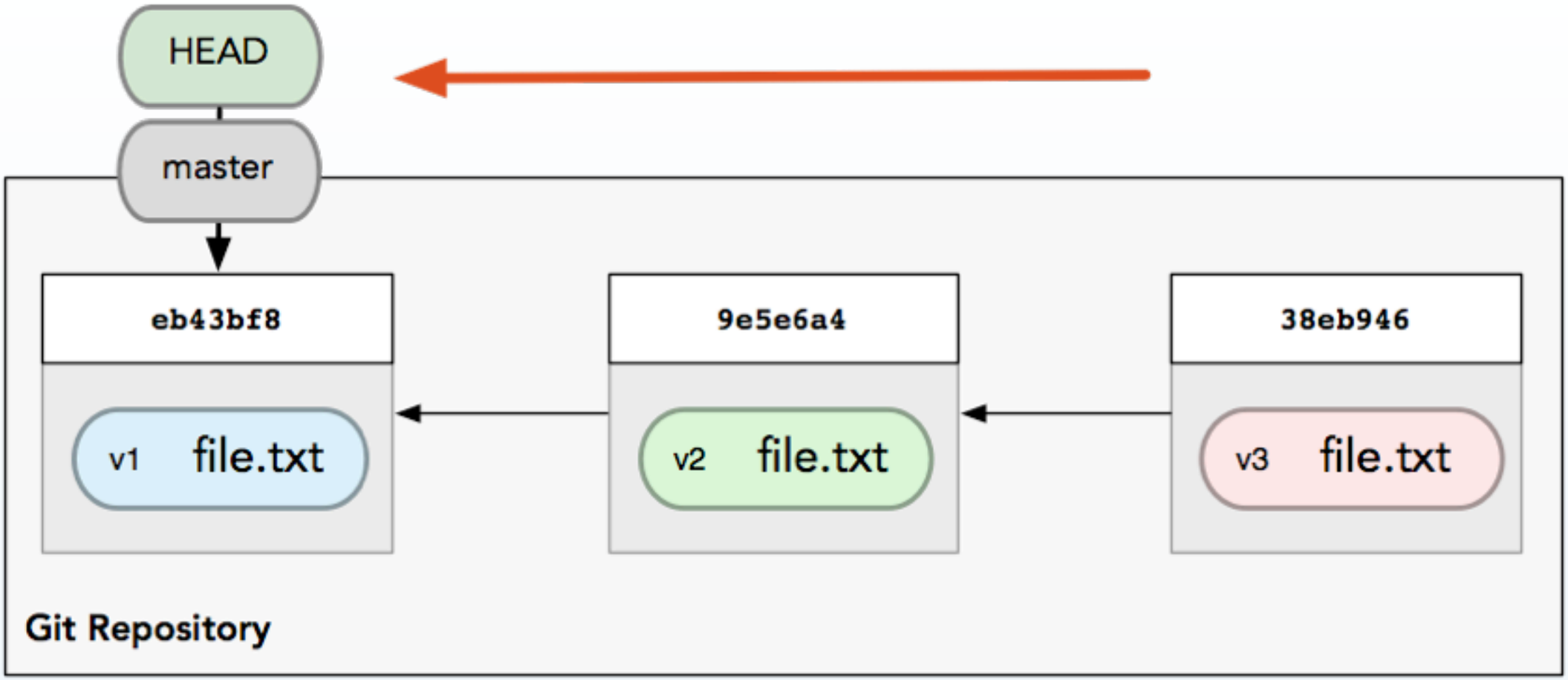


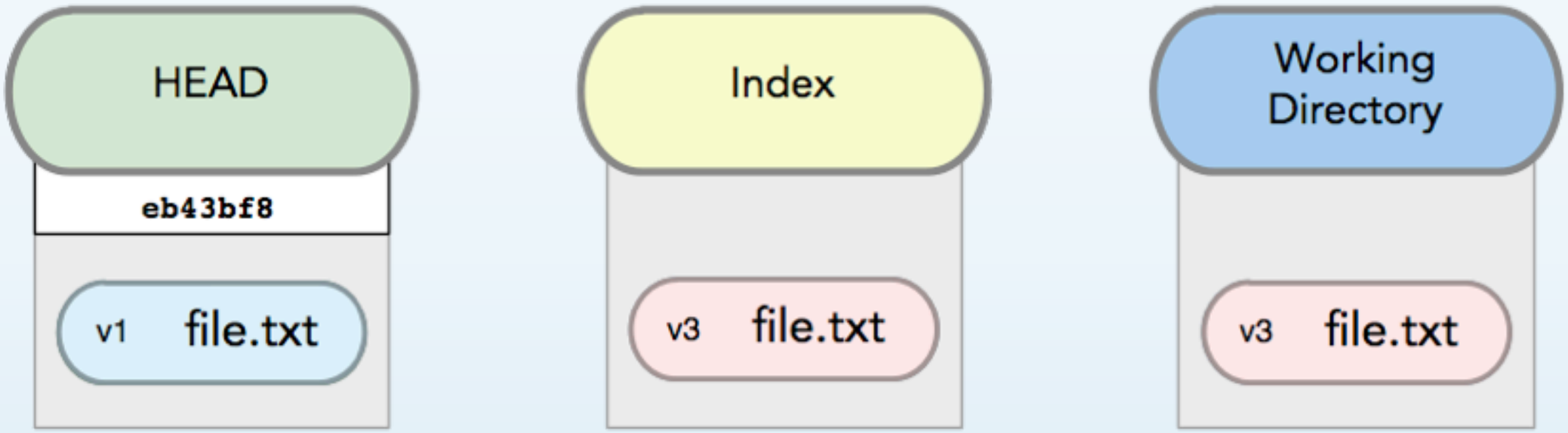

git commit -m ‘Yo soy muy inteligente’


HEAD
master
A
D2

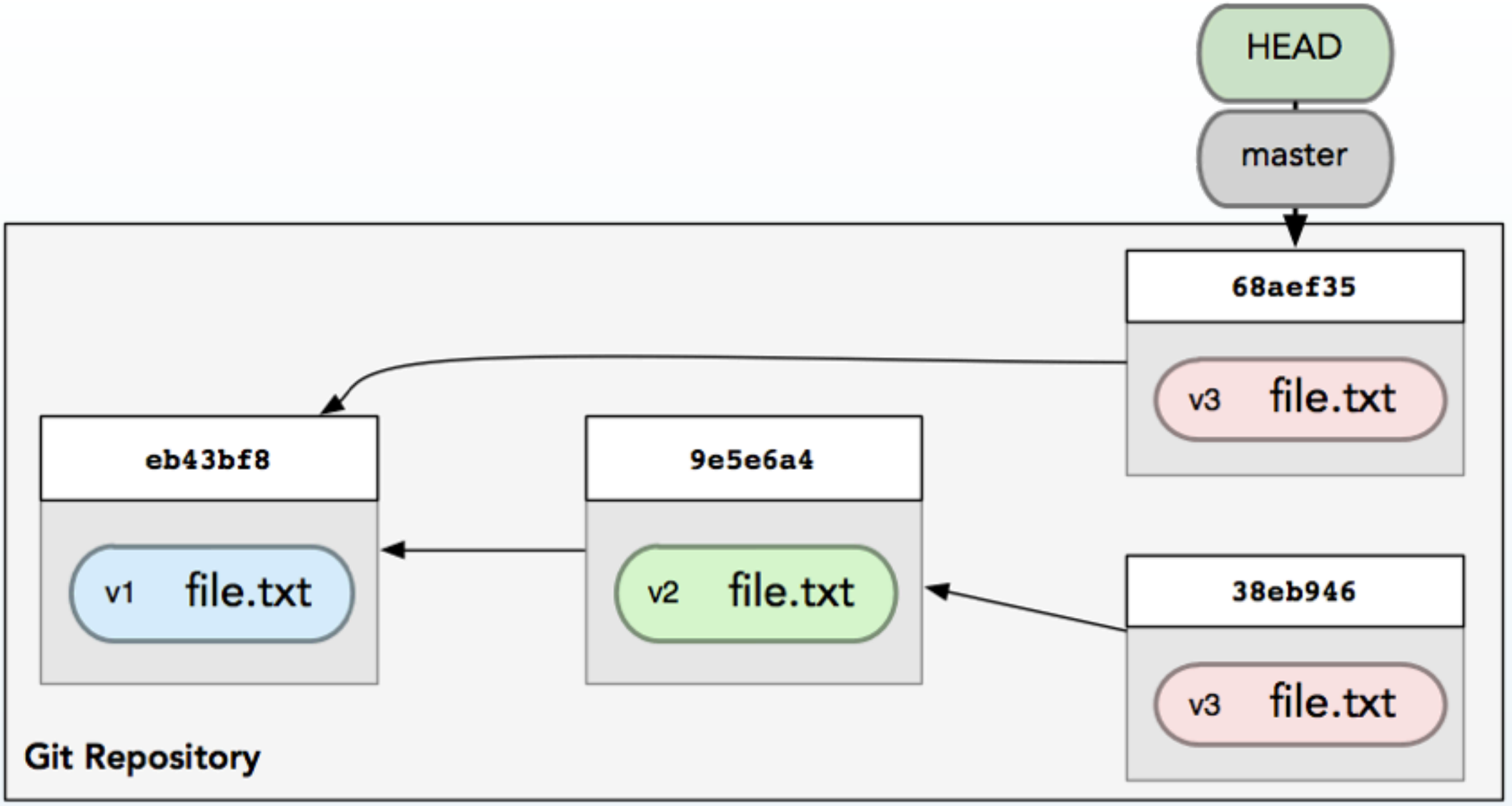

HEAD
master
eb43bf8
9e5e6a4
38eb946
v1 file.txt
v2 file.txt
v3 file.txt
Git Repository

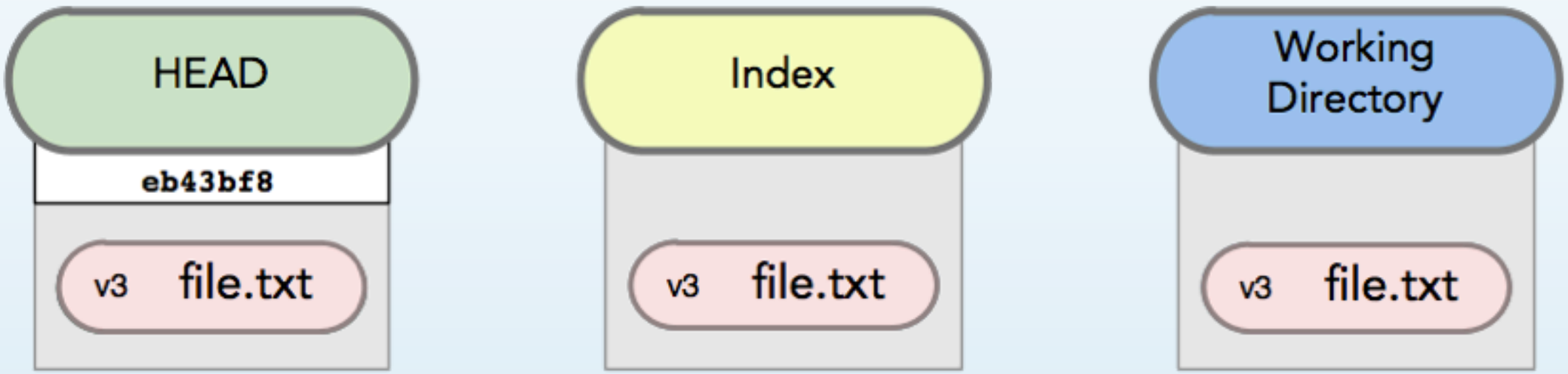

HEAD
38eb946
v3 file.txt
Index
v3 file.txt
Working Directory
v3 file.txt




# git reset --soft HEAD~2


HEAD
master
68aef35
v3 file.txt
eb43bf8
v1 file.txt
9e5e6a4
v2 file.txt
38eb946
v3 file.txt
Git Repository


HEAD
eb43bf8
v3 file.txt
Index
v3 file.txt
Working Directory
v3 file.txt

# git commit


SQUASH ALL THE THINGS!

```
commit 6eae70eee71446c4fb63d897c9feb62d62abc732
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Oct 31 09:01:27 2011 -0700

    wip: guárdelo en redis
```

```
commit 04c113569d5eec4fb53b5e35e953159a4c4449ca
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Oct 31 09:05:16 2011 -0700

    wip: hmm. guárdelo en cassandra

commit 6eae70eee71446c4fb63d897c9feb62d62abc732
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Oct 31 09:01:27 2011 -0700

    wip: guárdelo en redis
```

```
commit 71f6d906b763538d1a146cbf3934af6d19fdc348
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Oct 31 09:05:20 2011 -0700

    wip: maldita sea. tal vez riak?

commit 04c113569d5eec4fb53b5e35e953159a4c4449ca
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Oct 31 09:05:16 2011 -0700

    wip: hmm. guárdelo en cassandra

commit 6eae70eee71446c4fb63d897c9feb62d62abc732
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Oct 31 09:01:27 2011 -0700

    wip: guárdelo en redis
```

```
commit 8c9edaee51647f392e20199bfc1bb15c9e221a1a
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Oct 31 09:05:20 2011 -0700

    wip: por que coño, riak? ahora, mongodb...

commit 71f6d906b763538d1a146cbf3934af6d19fdc348
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Oct 31 09:05:20 2011 -0700

    wip: maldita sea. tal vez riak?

commit 04c113569d5eec4fb53b5e35e953159a4c4449ca
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Oct 31 09:05:16 2011 -0700

    wip: hmm. guárdelo en cassandra

commit 6eae70eee71446c4fb63d897c9feb62d62abc732
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Oct 31 09:01:27 2011 -0700

    wip: guárdelo en redis
```

```
commit 6eae70eee71446c4fb63d897c9feb62d62abc732
Author: Scott Chacon <schacon@gmail.com>
Date:   Mon Oct 31 10:01:27 2011 -0700

    solución perfecta en el primer intento: mongodb
```

# WIP

# </reset>

# Certificate of Achievement

This to certify that

you people

has successfully completed

Git Reset Training

Scott "Dragon" Chacon

Course Manager

# CODA

## en resúmen

# funciones de los árboles

# funciones de los árboles

**HEAD** el último commit, generador del proximo

# funciones de los árboles

**HEAD** el último commit, generador del proximo

**Índice** el proximo commit

# funciones de los árboles

**HEAD** el último commit, generador del proximo

**Índice** el proximo commit

**Dir de Trabajo** caja de arena

# Muchas gracias!

# **scott** chacon

github.com/schacon/tres-arboles

**@chacon**