Arquitectura
Avanzada
de
Aplicaciones

NETT FORMACIÓN

SCM: GIT

BB.DD. – Presentación -

es.linkedin.com/in/noelmd/

noel.nmd@gmail.com

GIT - Introducción

NETT FORMACIÓN

GIT - Introducción

GIT - Introducción

Source Code Management

GIT - Introducción

NETT FORMACIÓN

GIT - Introducción

NETT FORMACIÓN

GIT - Introducción

NETT FORMACIÓN

First of all...

Clear your mind

Git origin

It was born in 2005, developed by the kernel linux team, leaded by Linus Torvalds.

Tries to improve BitKeeper.

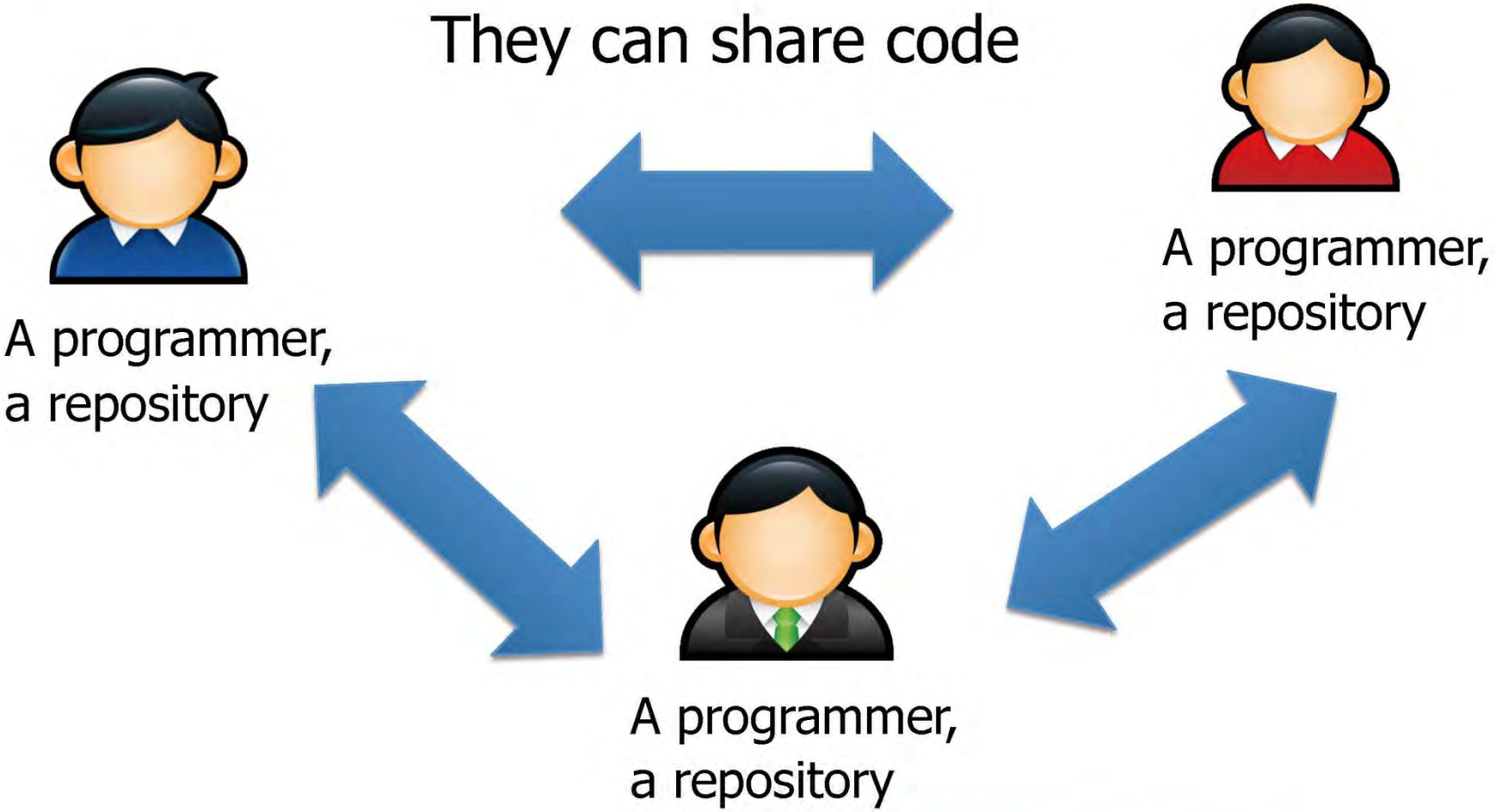
Git origin

It was born in 2005, developed by the kernel linux team, leaded by Linus Torvalds.

Tries to improve BitKeeper.

What is a DVCS

They can share code



Everyone has a local repository

Git: Goals

- Fast
- Simplicity
- Multi Branches
- Distributed
- Able to manage big projects

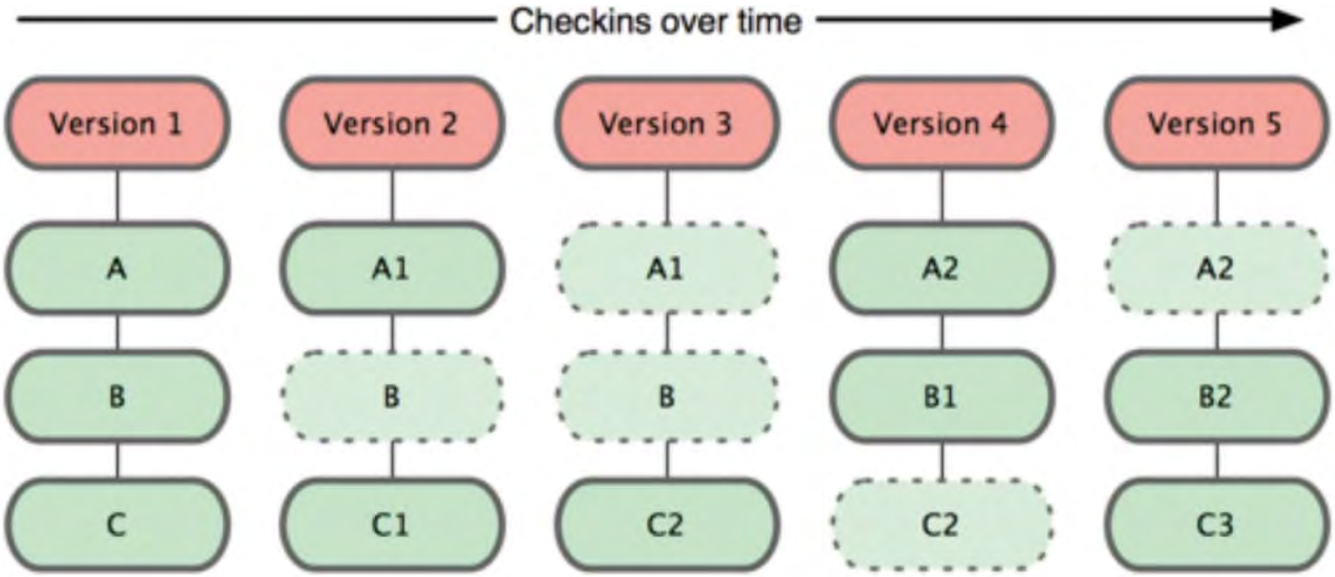
NETT FORMACIÓN

Git: Set of snapshots

Takes a pictur like at that moment and s napshot,

NETT FORMACIÓN

Git: Set of snapshots



Takes a picture of what all your files look like at that moment and stores a reference to that snapshot,

NETT FORMACIÓN

Every operation is done in your local repository.

You don’t need connection with some other server.

You can work offline without problems

Faster than centralized repositories

git Integrity

NETT FORMACIÓN

Everything is identified by hashes.

It is impossible to change the content of any file or directory without Git knowing about it.

Can't lose information or get file corruption without Git being able to detect it.

Checksum (sha1):

24b9da6552252987aa493b52f8696cd6d3b00373

Tracked files are files that were in the last snapshot; they can be unmodified, modified, or staged.

Untracked files are everything else.

Patterns

#: Comments

!: Negates the pattern

foo/: Directory named foo (not a file foo)

*: Everything

.html , !foo.html, /.js

NETT FORMACIÓN

Patterns

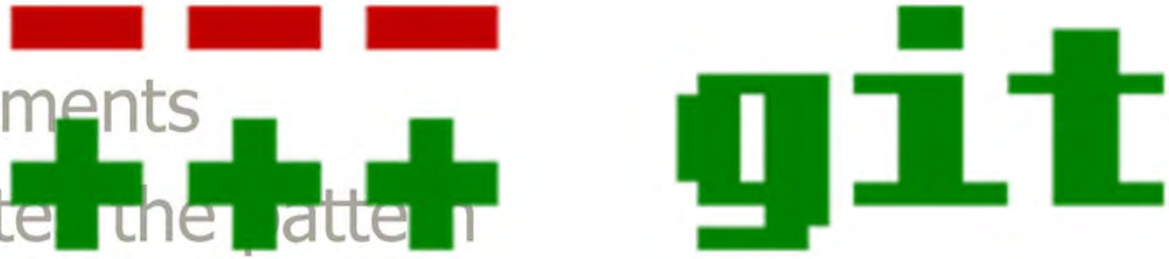
#: Comments

!: Negate the pattern

foo/: Directory named foo (not a file foo)

*: Everything

.html , !foo.html, /.js



/ignore

https://github.com/github/gitignore

git Adding .gitignore

NETT FORMACIÓN

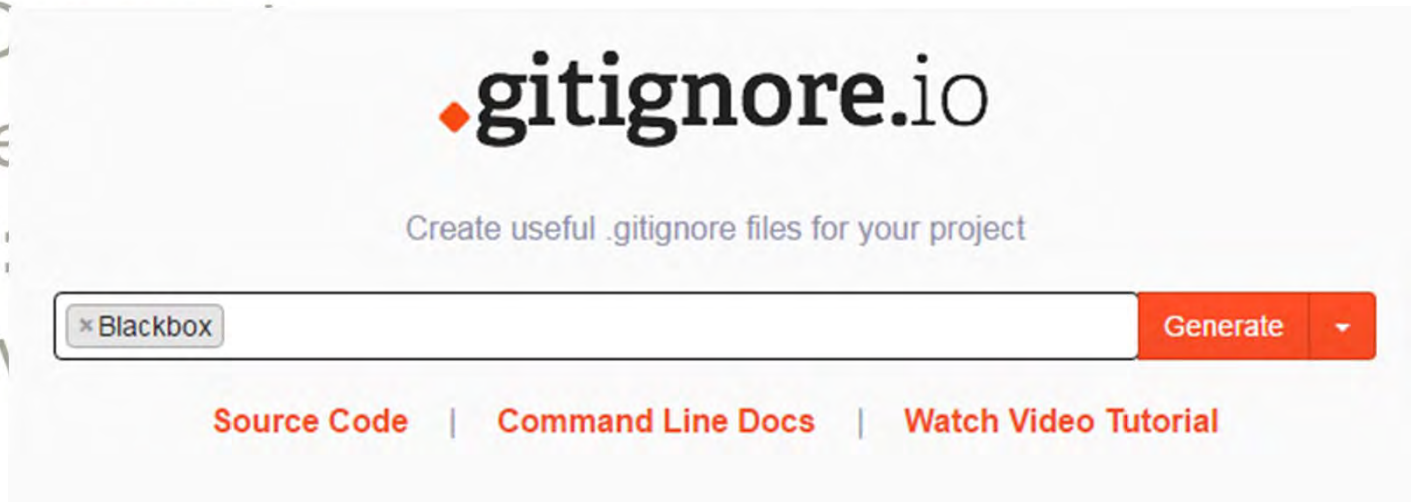
Patterns

#: C

!: Ne

foo/

*: Ev



.html , !foo.html, /.js

https://www.gitignore.io/

git Repository Types

There are two different types of Git repos:

- **Bare:** Has no working directory. Not be used for normal development. No direct commits.

- **Development:** Typical repository. Maintains current branch, provides checked-out copy of the current branch in a **working directory**

Repository Types

There are two different types of Git repos:

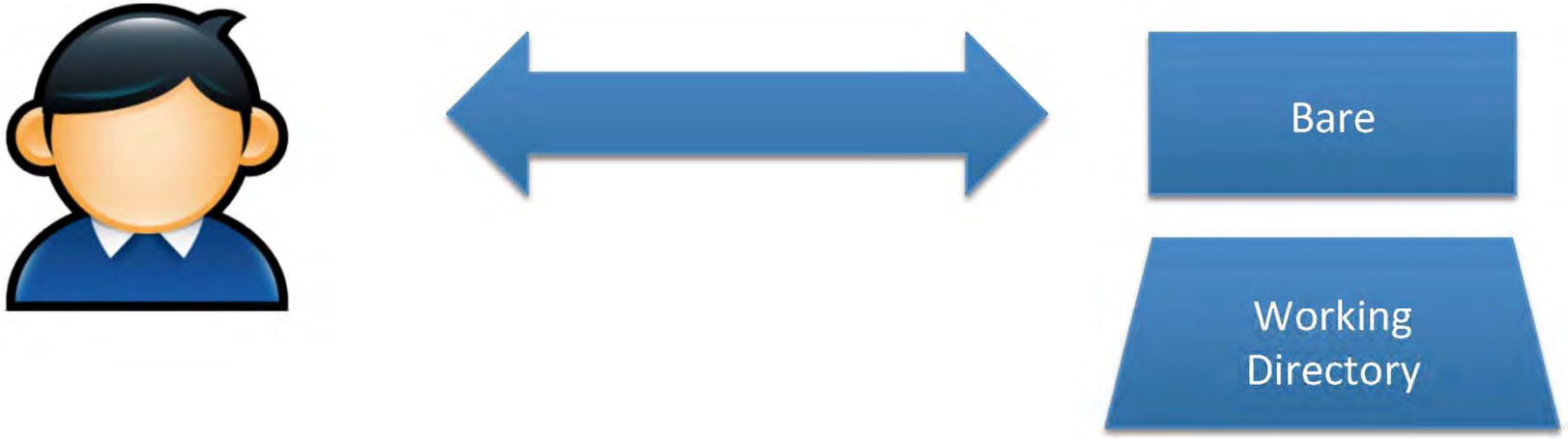
- **Bare:** Has no working directory. Not be used for normal development. No direct commits.

- **Development:** Typical repository. Maintains current branch, provides checked-out copy of the current branch in a **working directory**

git Repository Types

Bare repo is crucial for collaborative development.

Other developers clone and fetch from the bare repository and push updates to it



Bare repo creation

```
$ git init <repository> --bare
```

Best Practice:

A published repository should be bare

GIT - Repos

NETT FORMACIÓN

Create your first repository

1) Create a specific folder where your repository is going to be created.

2) $ git init

Create a repository

You can create or clone a Git repository:

- Create

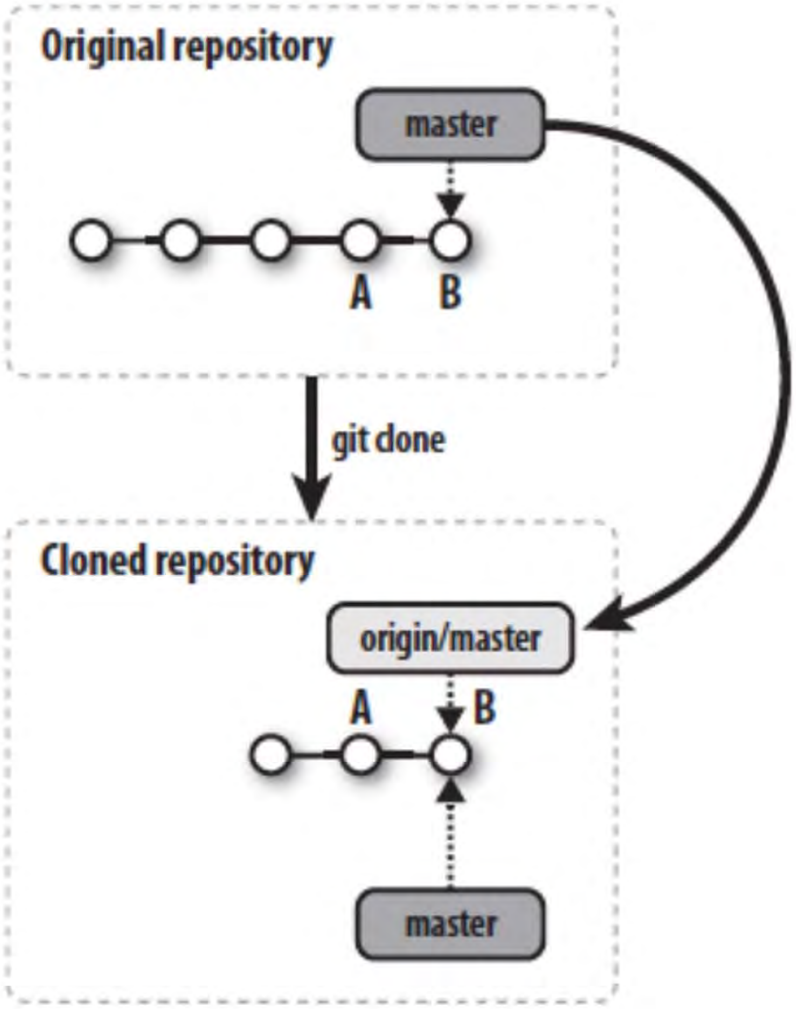
```
$ git init
```

- Clone

```
$ git clone <remote-url>
```

Create a repository

Clone a repo:



Create a repository

```
Noel@BlackPearl MINGW64 /l
$ mkdir Repo_AAA

Noel@BlackPearl MINGW64 /l
$ cd Repo_AAA/

Noel@BlackPearl MINGW64 /l/Repo_AAA
$ git init
Initialized empty Git repository in L:/Repo_AAA/.git/

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$
```

GIT - Configuración

git config --global user.name “My Name“

git config --global user.email
my_email@whatever.com

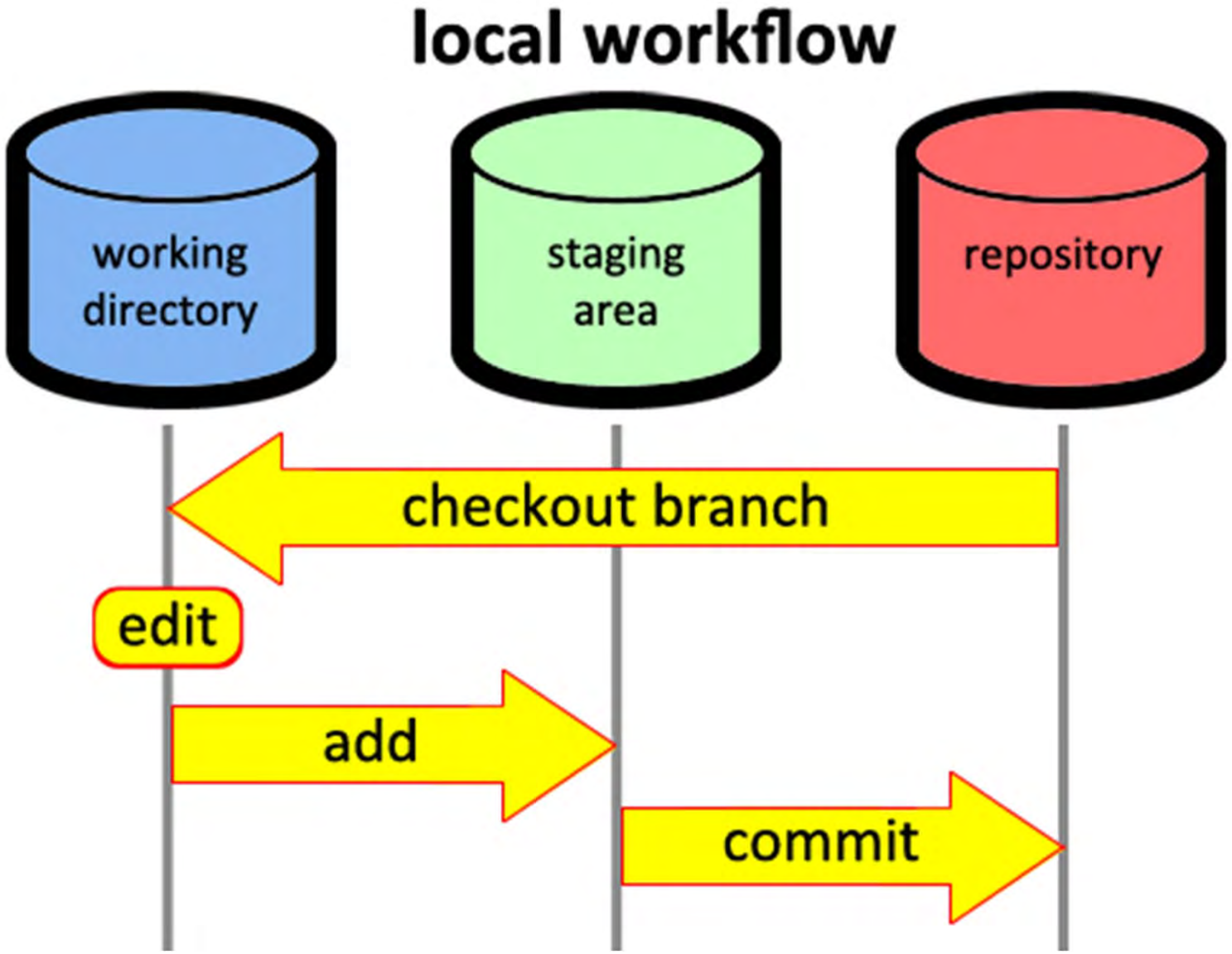
git config --global core.autocrlf true

git config --global core.safecrlf true

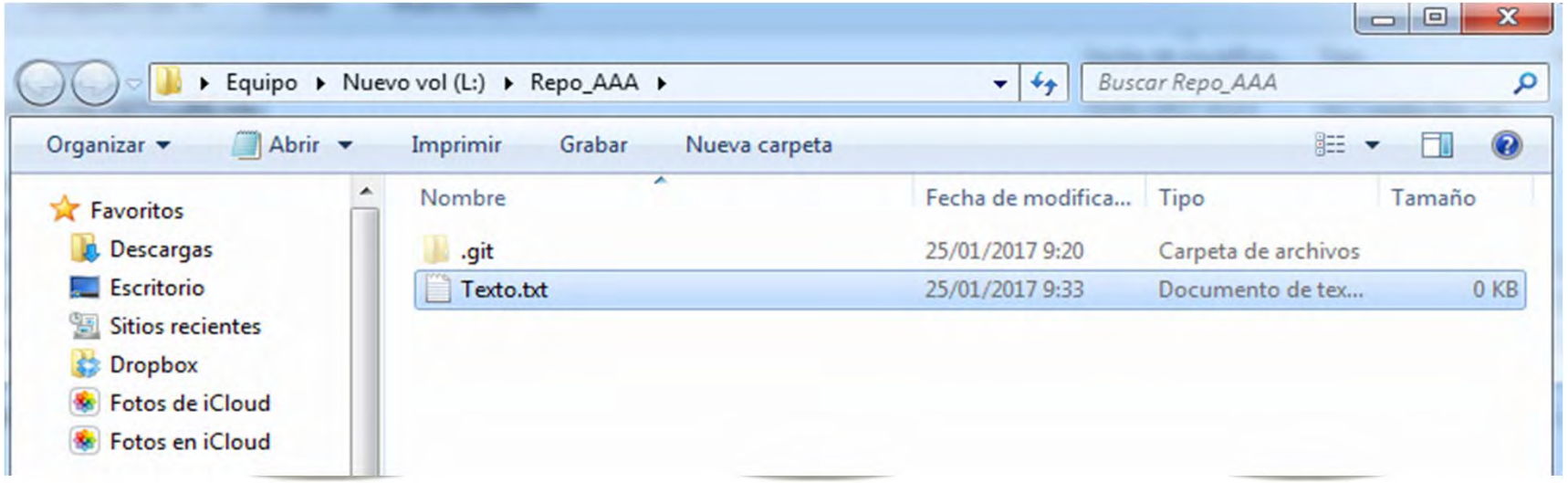
GIT - Configuración

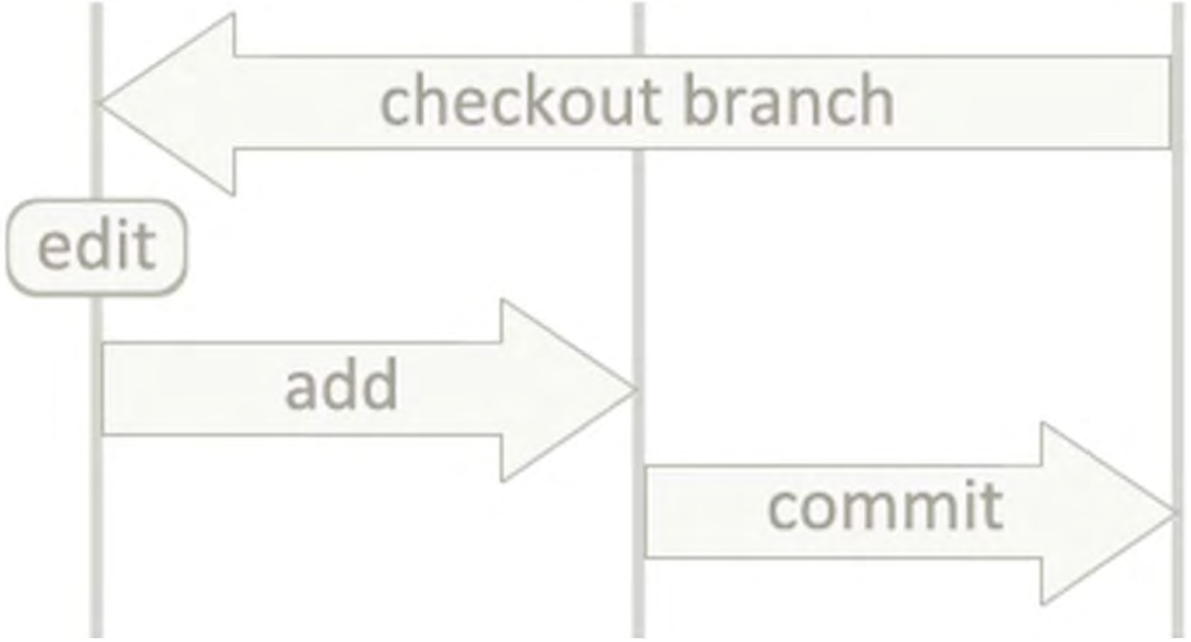

```
git config --global alias.co checkout
git config --global alias.ci commit
git config --global alias.st status
git config --global alias.br branch
git config --global alias.hist "log --
pretty=format:'%h %ad | %s%d [%an]' --graph --
date=short"
git config --global alias.type 'cat-file -t'
git config --global alias.dump 'cat-file -p'
```

GIT – Área de Trabajo

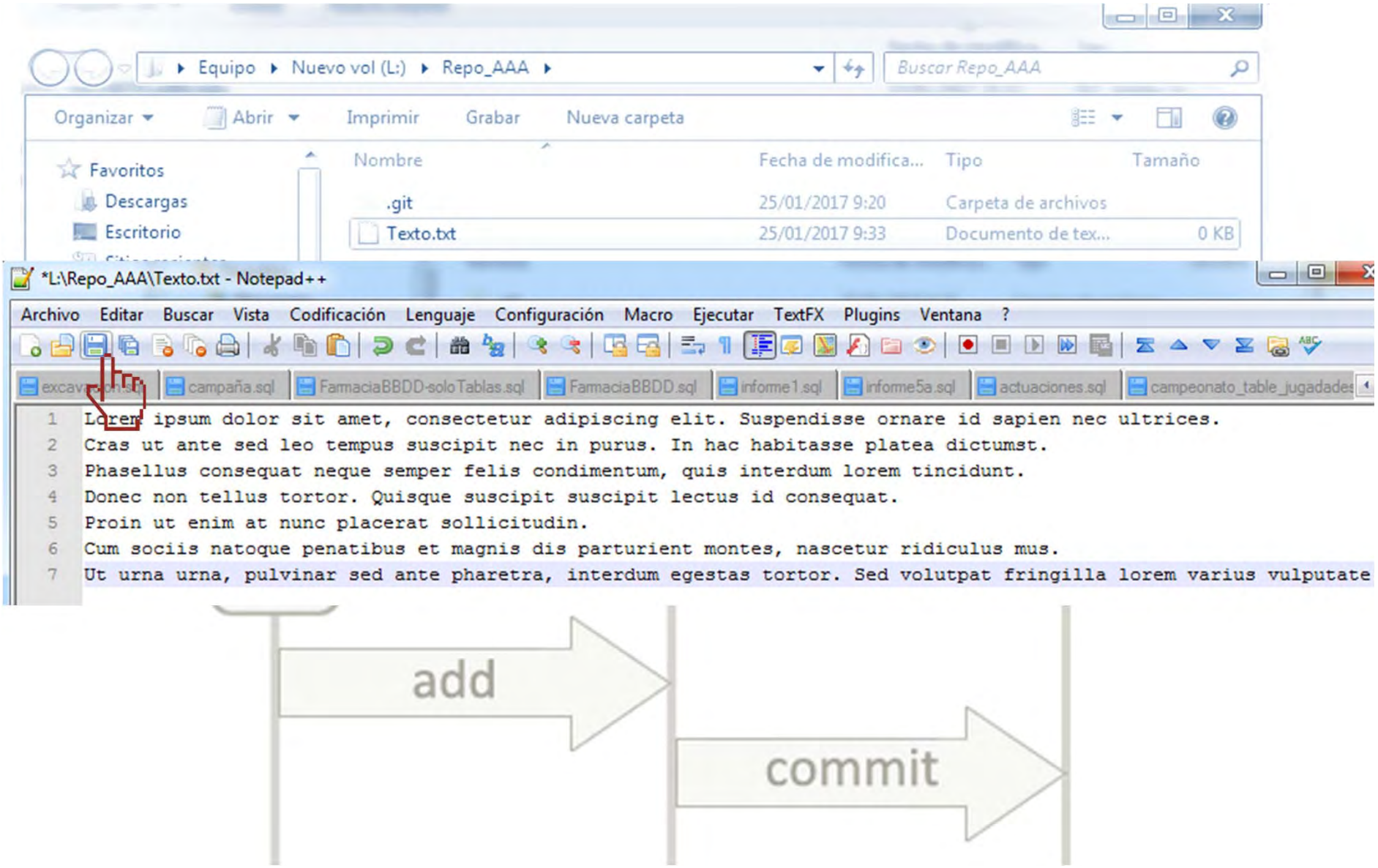


GIT – Área de Trabajo

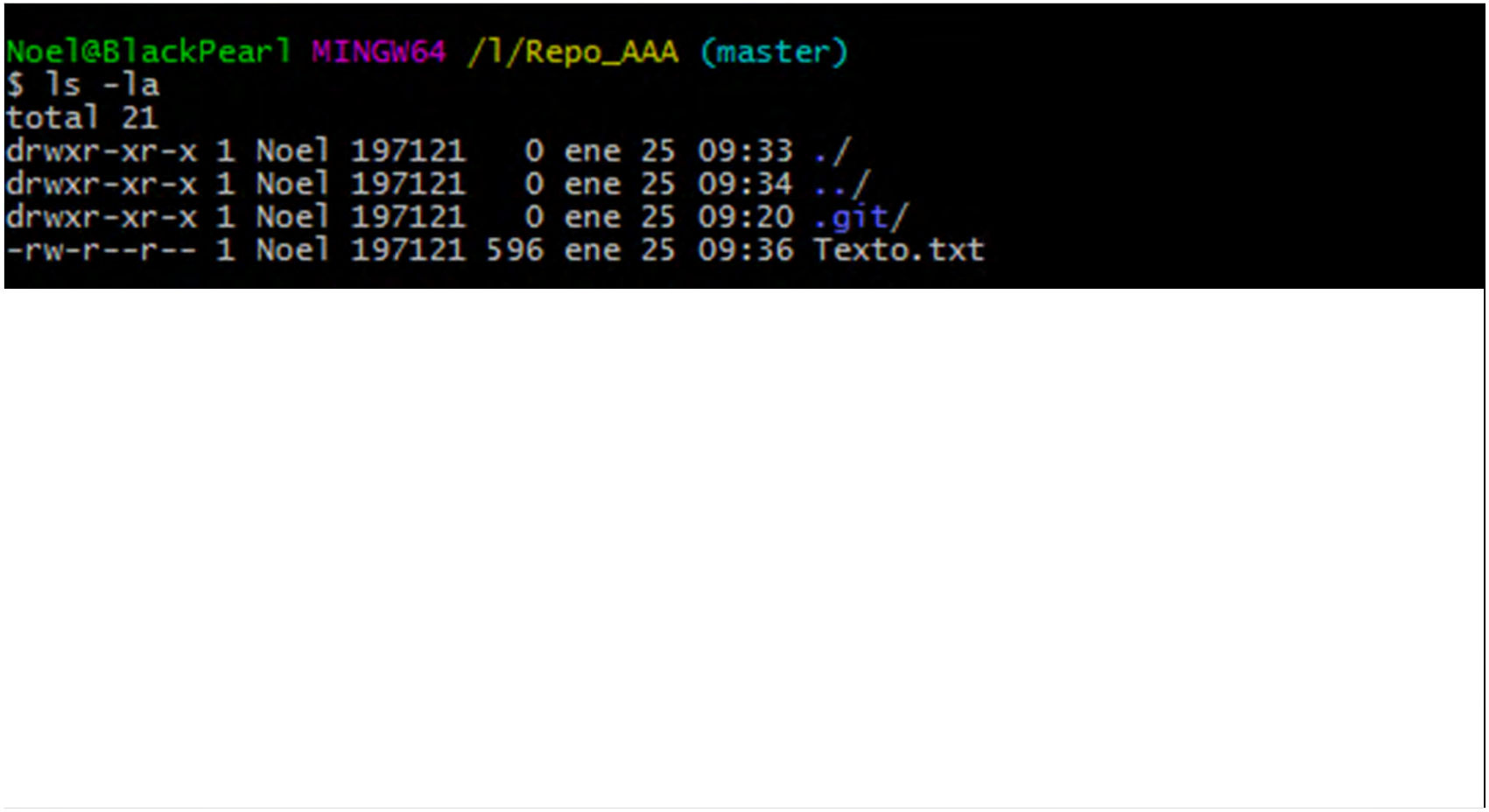




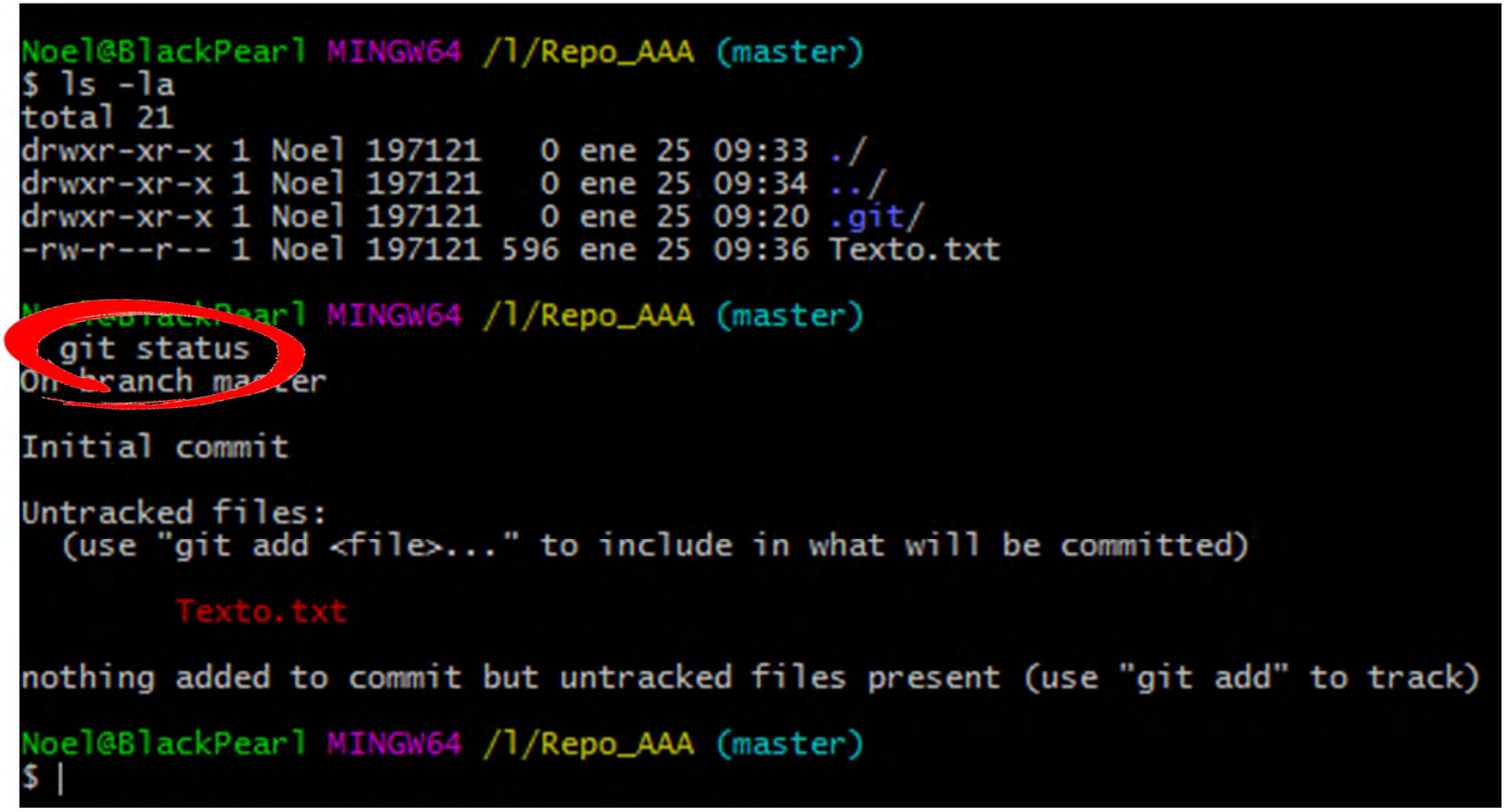
GIT – Área de Trabajo



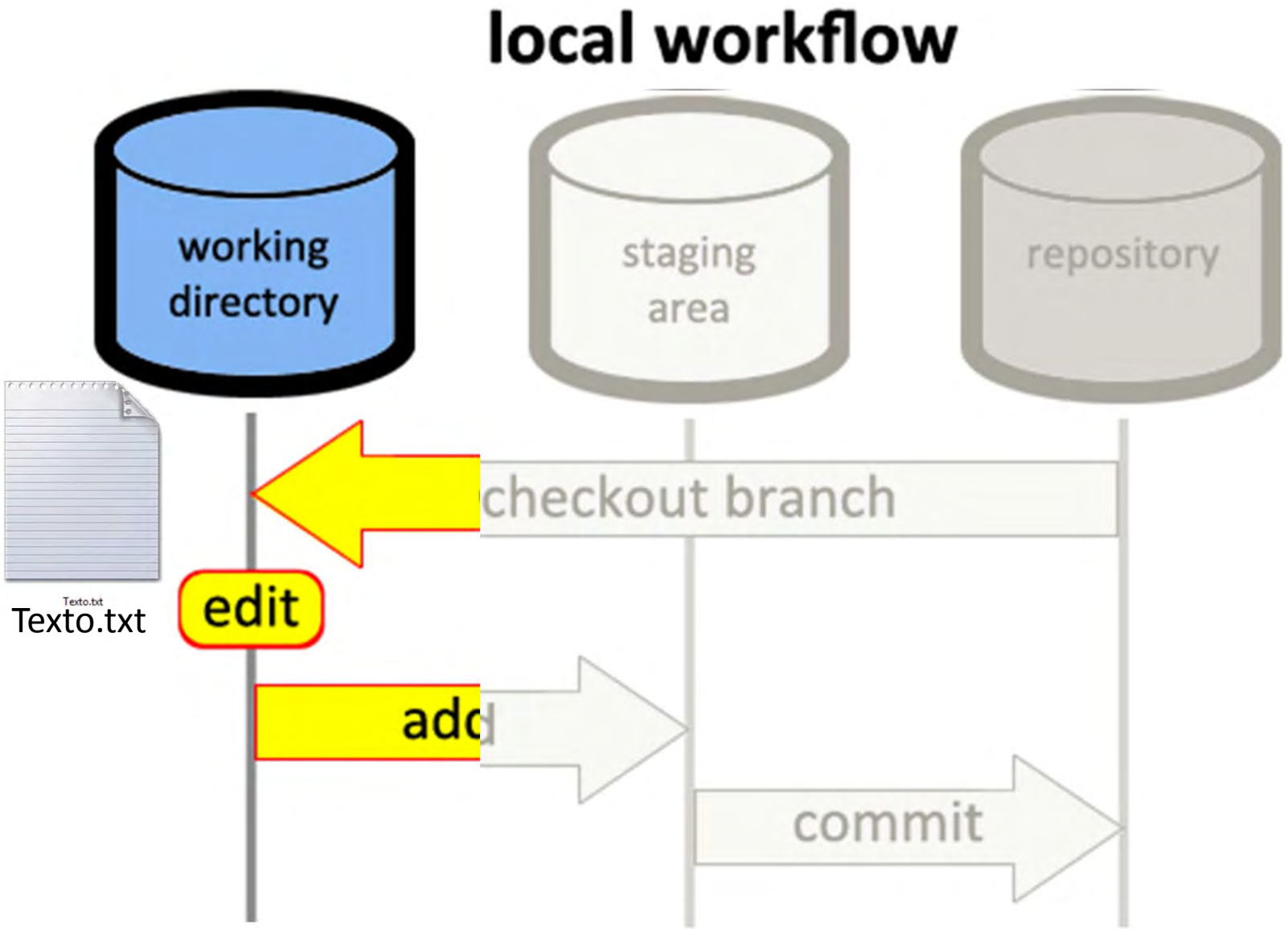
GIT – Área de Trabajo



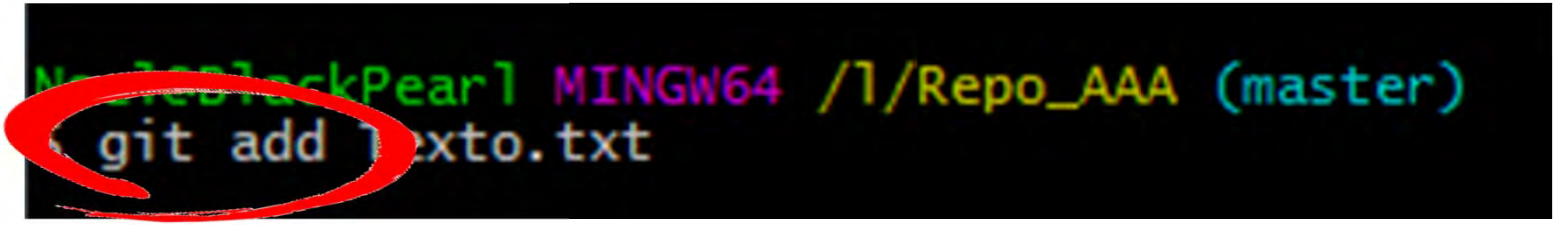
GIT – Área de Trabajo



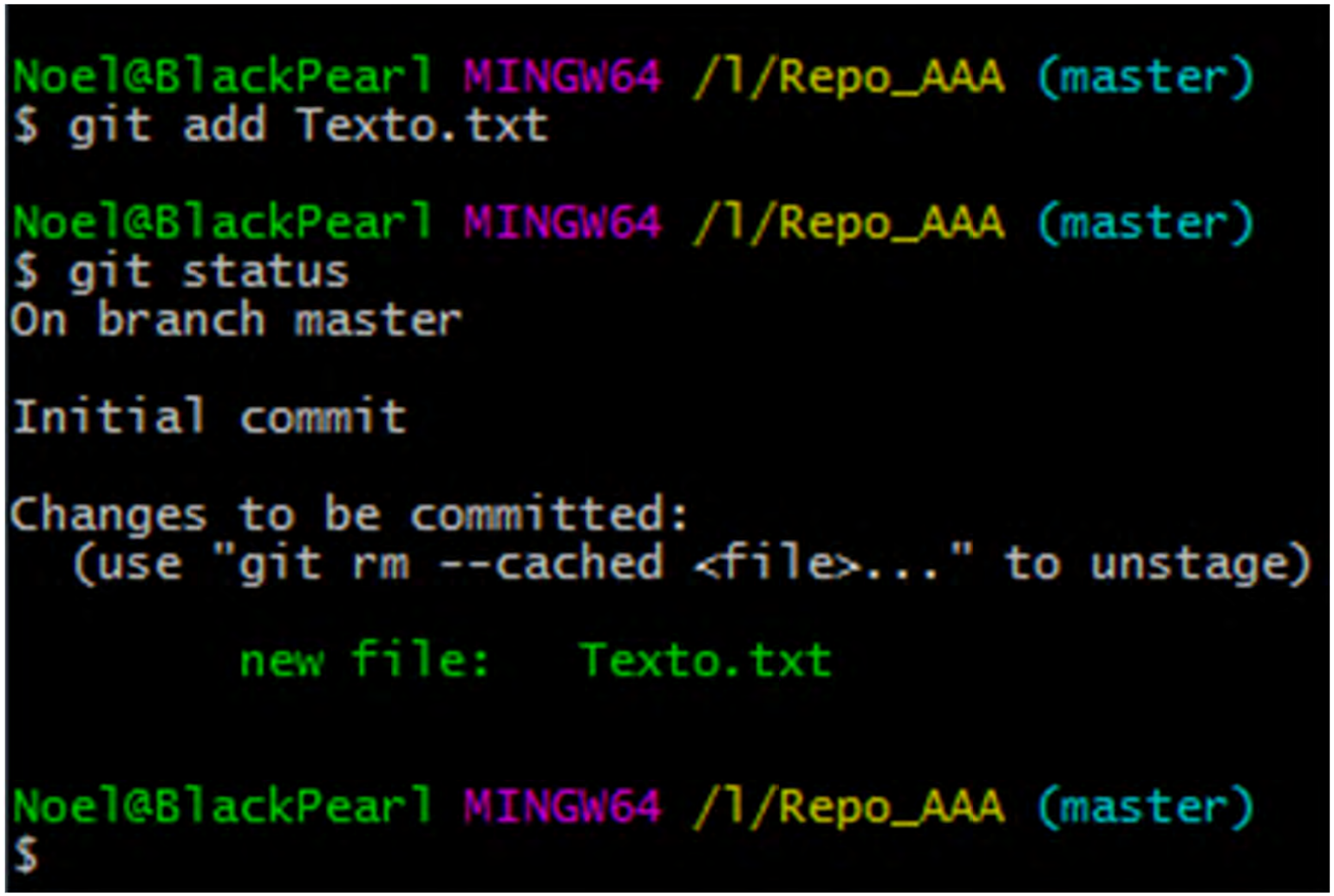
GIT – Área de Trabajo



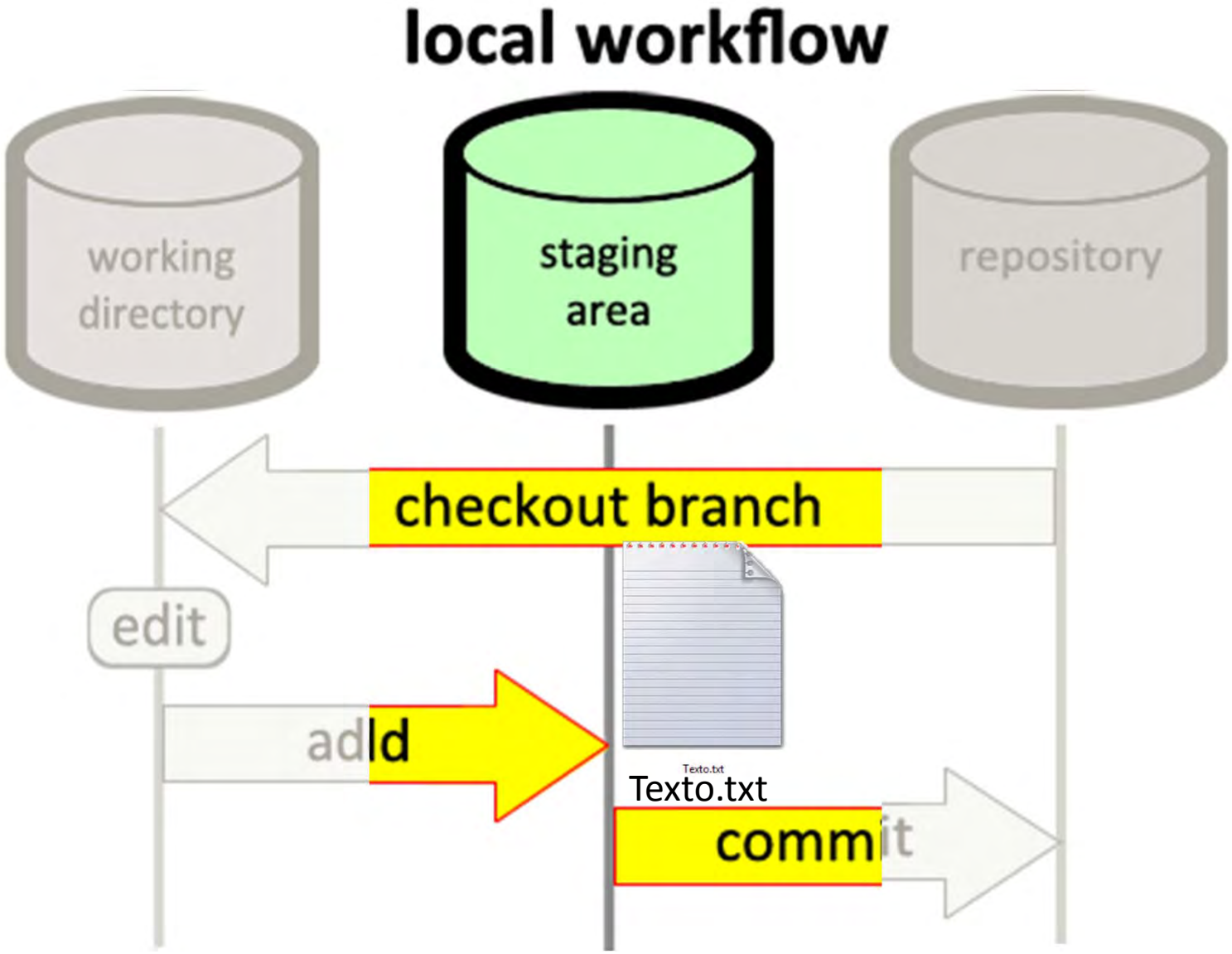
GIT – Área de Trabajo



GIT – Área de Trabajo



GIT – Área de Trabajo



GIT – Área de Trabajo

NETT FORMACIÓN

COMMIT/

A commit is used to record changes to a repository.

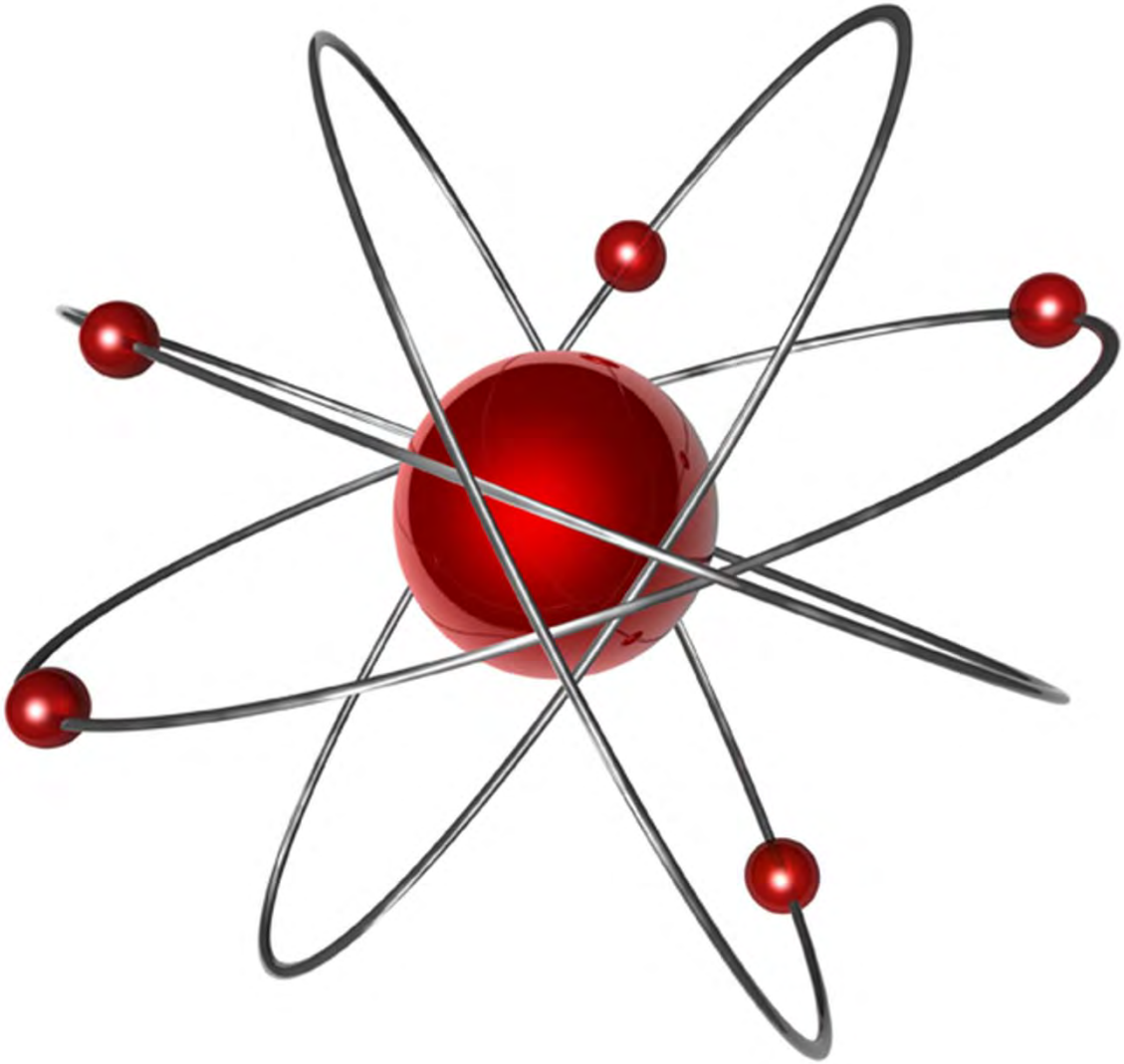
When a commit occurs, Git records a “snapshot” of the index and places that snapshot in the object store.

NETT FORMACIÓN

A commit is the only method o introducing changes to a repository.

Commits are often introduced explicitly by a developer but Git can introduce commits (as in a merge).

Atomic changesets



Every Git commit represents a single, atomic changeset with respect to the previous state.

Every changes apply in a commit or none (atomicity).

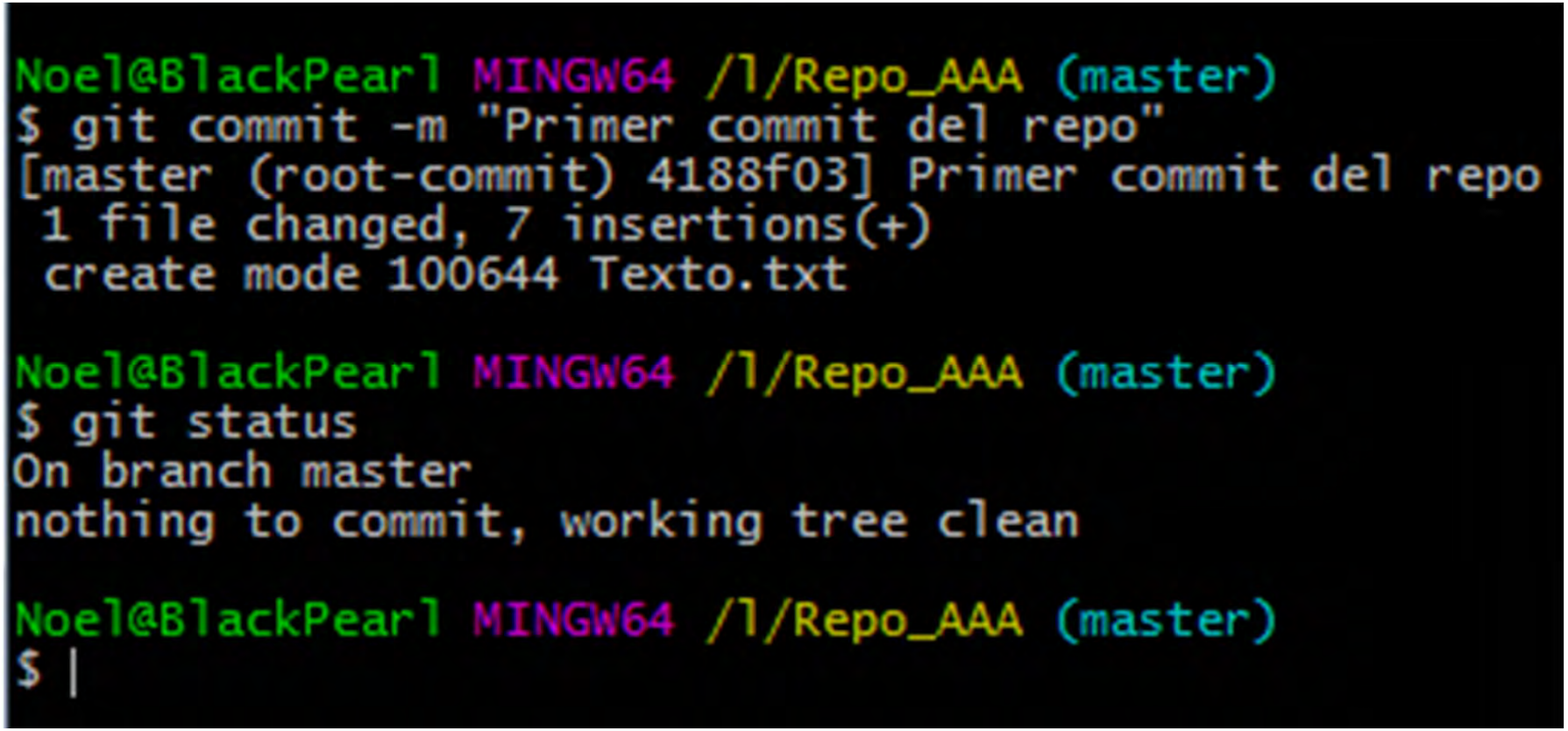
You can be assured that Git has not left your repository in some transitory state between one commit snapshot and the next.

Every Git commit is a hex-digit SHA1 ID.

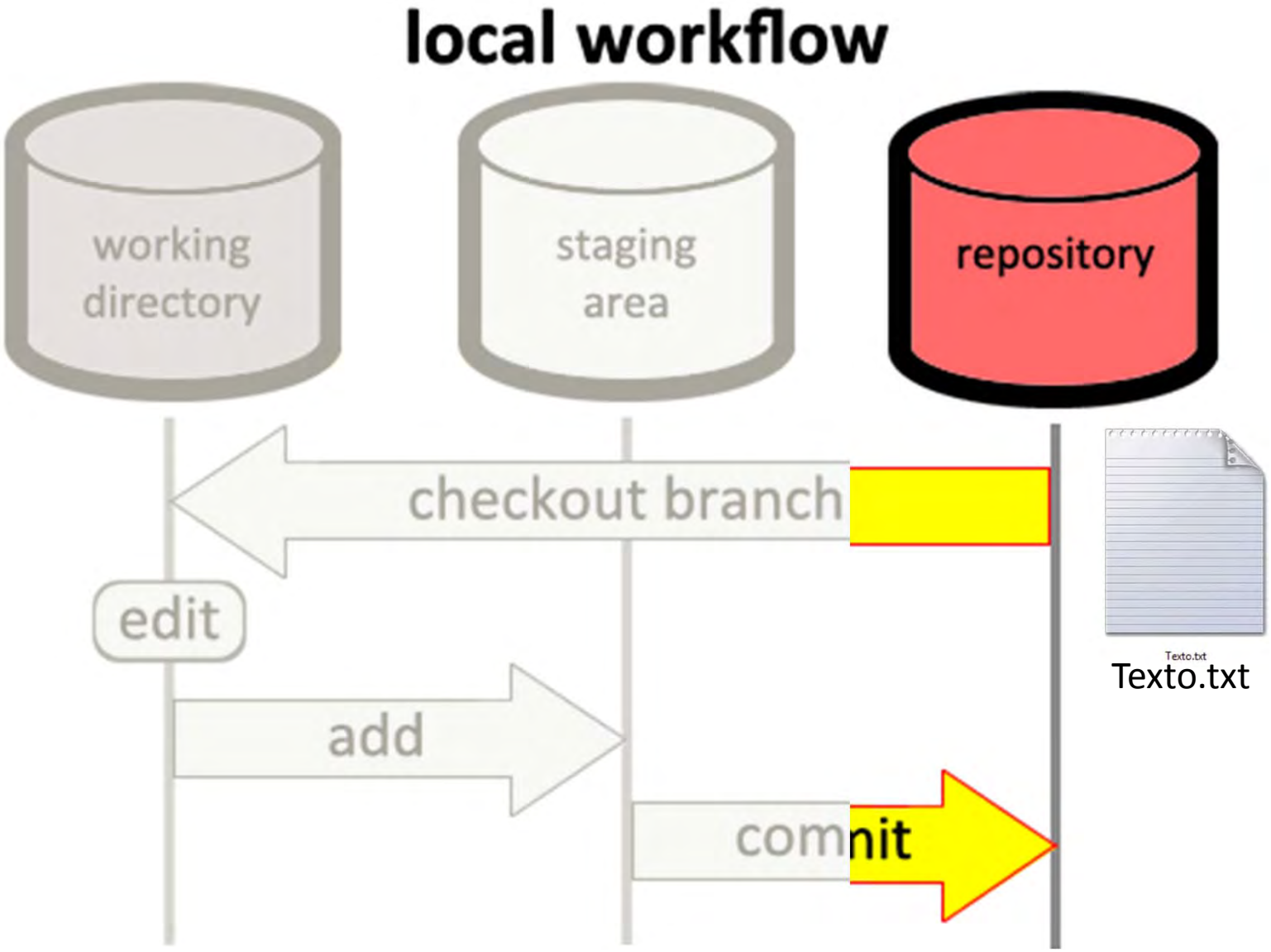
Each commit ID is globally unique—not just for one repository, but for any and all repositories

GIT – Área de Trabajo

GIT – Área de Trabajo



GIT – Área de Trabajo



NETT FORMACIÓN

2. Commit

```
$ git commit –m "message"
```

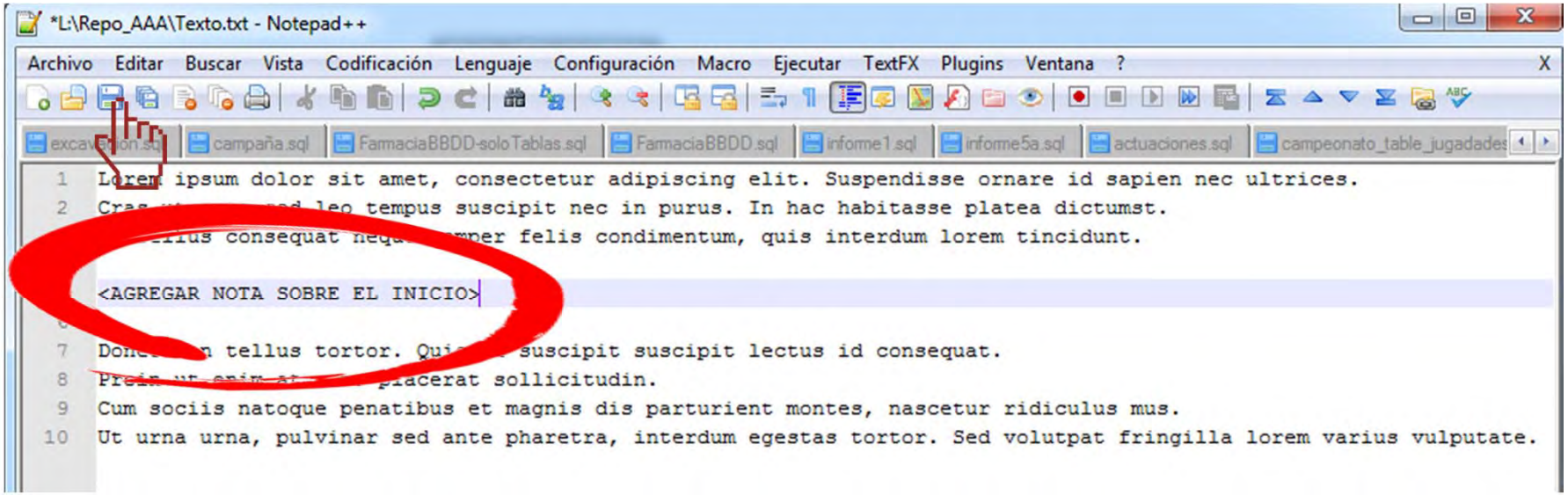
If the file is not managed by the repo:

```
$ git commit –am "message"
```

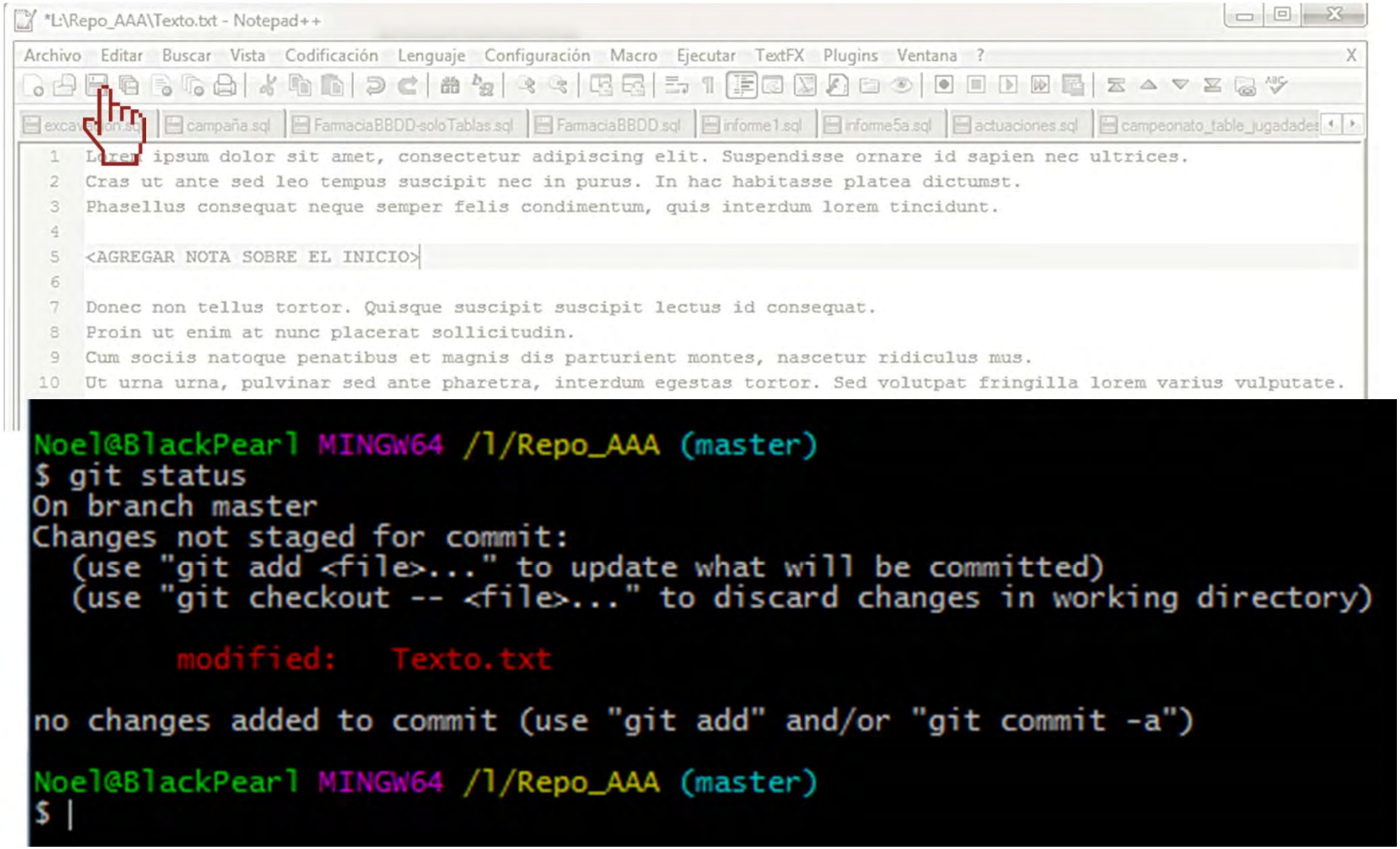
NETT FORMACIÓN

```
$ git commit -m “Message”

$ git commit -am “Message”

$ git commit -m “Message” <file>

$ git commit --amend
```

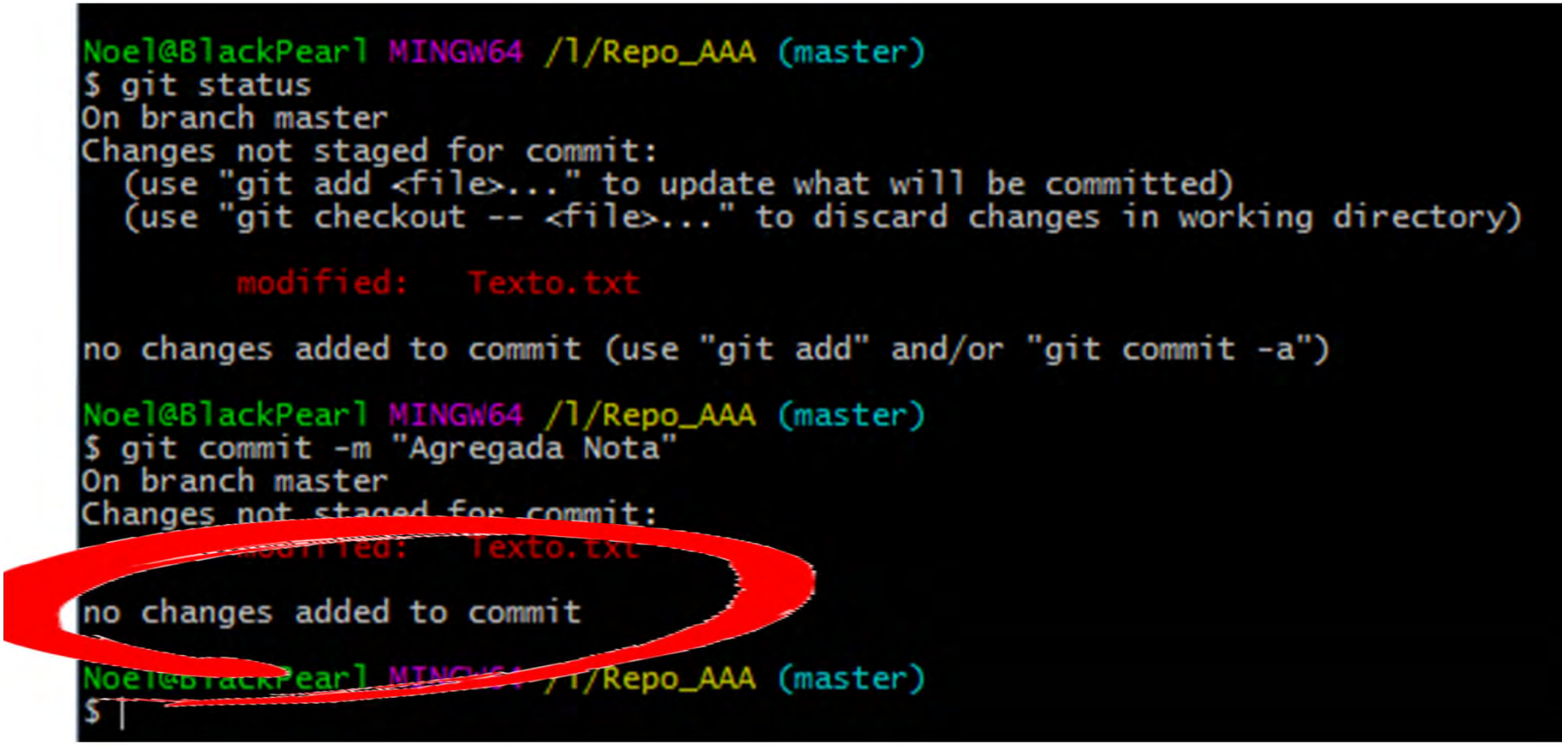
GIT – Área de Trabajo



GIT – Área de Trabajo



GIT – Área de Trabajo

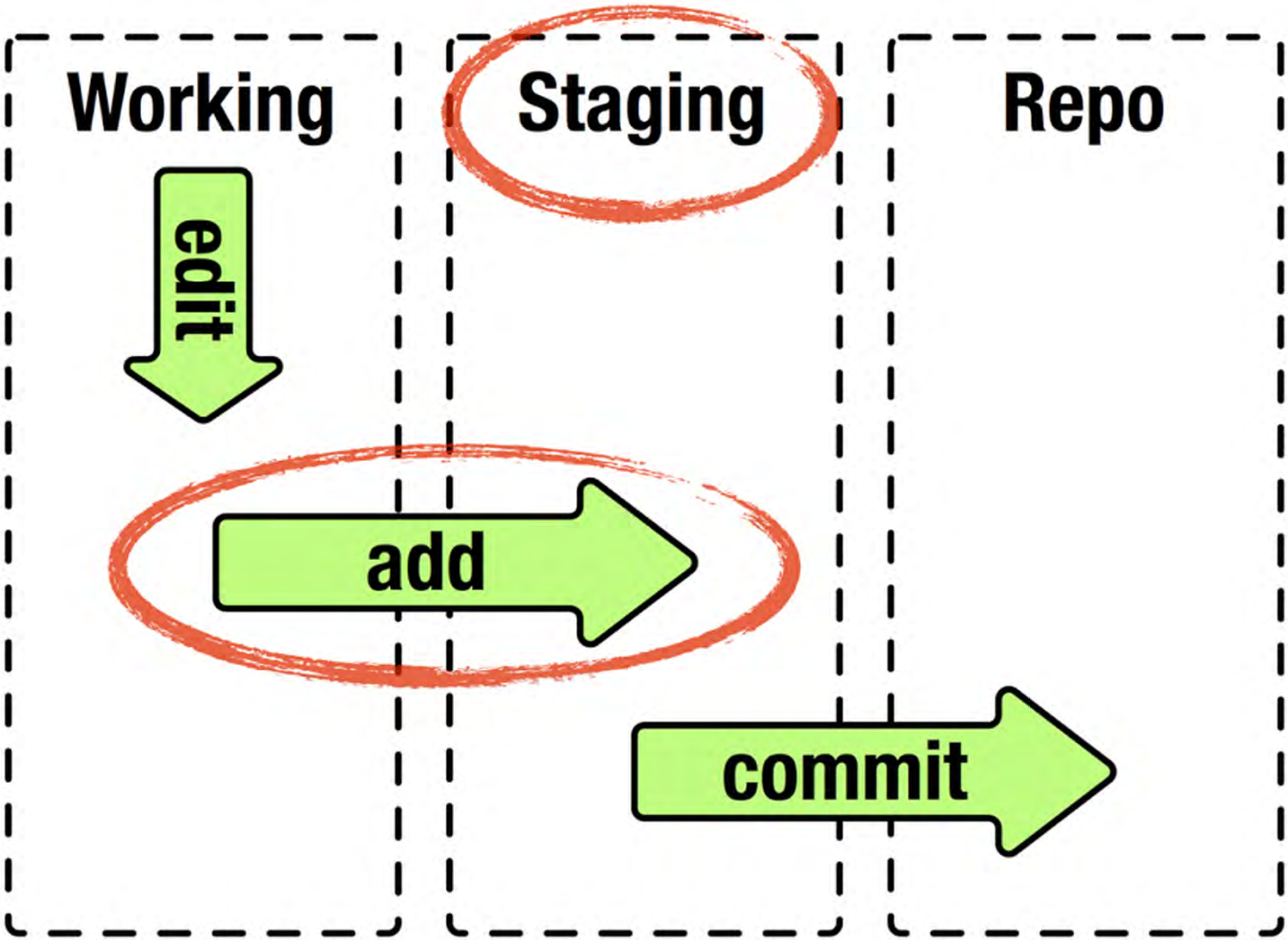


GIT – Área de Trabajo

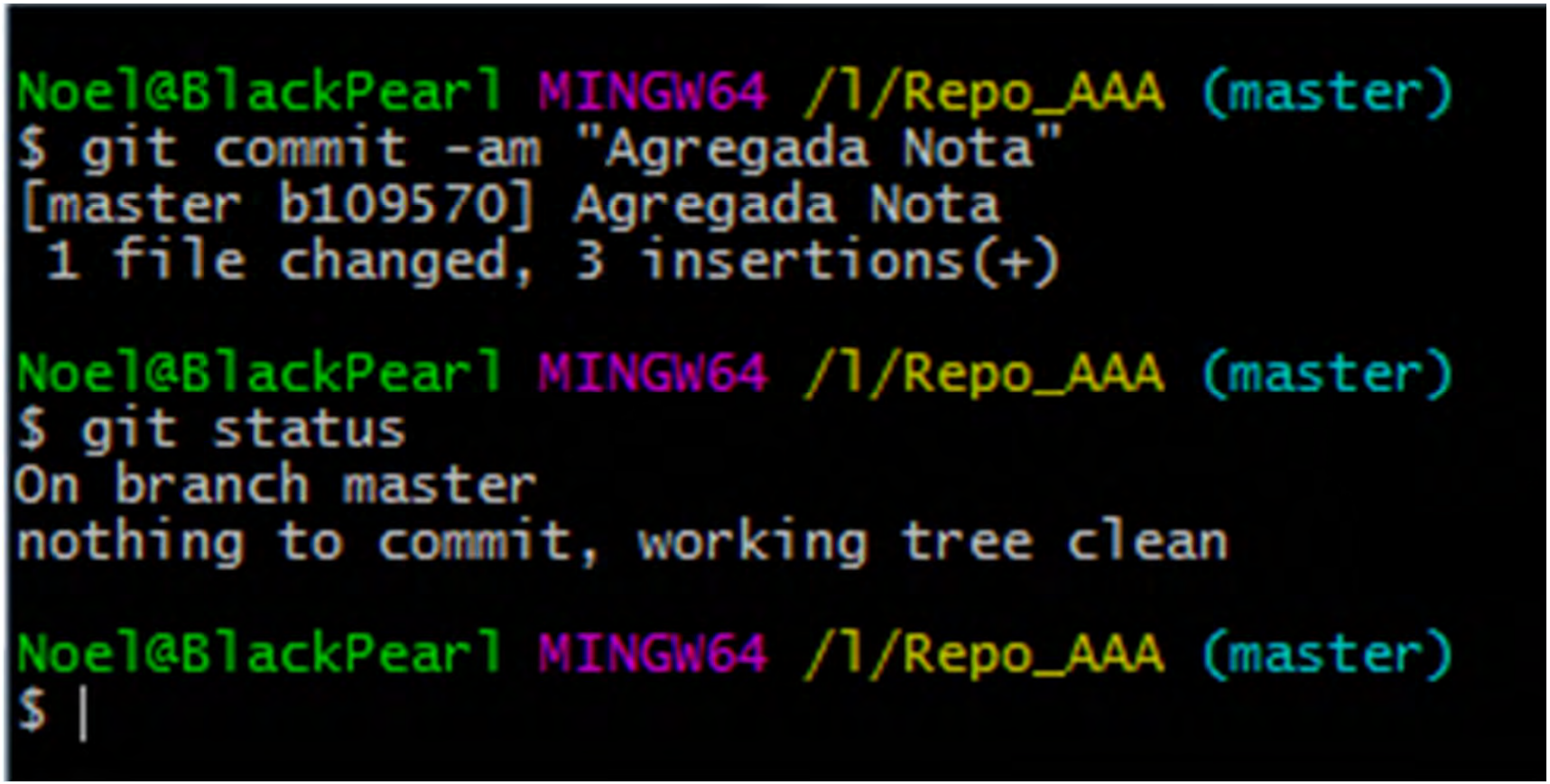
NETT FORMACIÓN

COMMIT

GIT – Área de Trabajo



GIT – Área de Trabajo



NETT FORMACIÓN

3. Remove a file from the staging area

#Before first commit

$ git rm --cached <file>

#After first commit

$ git reset HEAD <files>

Basic Flow: First Commit

3. Remove a file from the staging area

#Before first commit

$ git rm --cached <file>

#After first commit

$ git reset HEAD <files>

NETT FORMACIÓN

3. Remove a file from the staging area

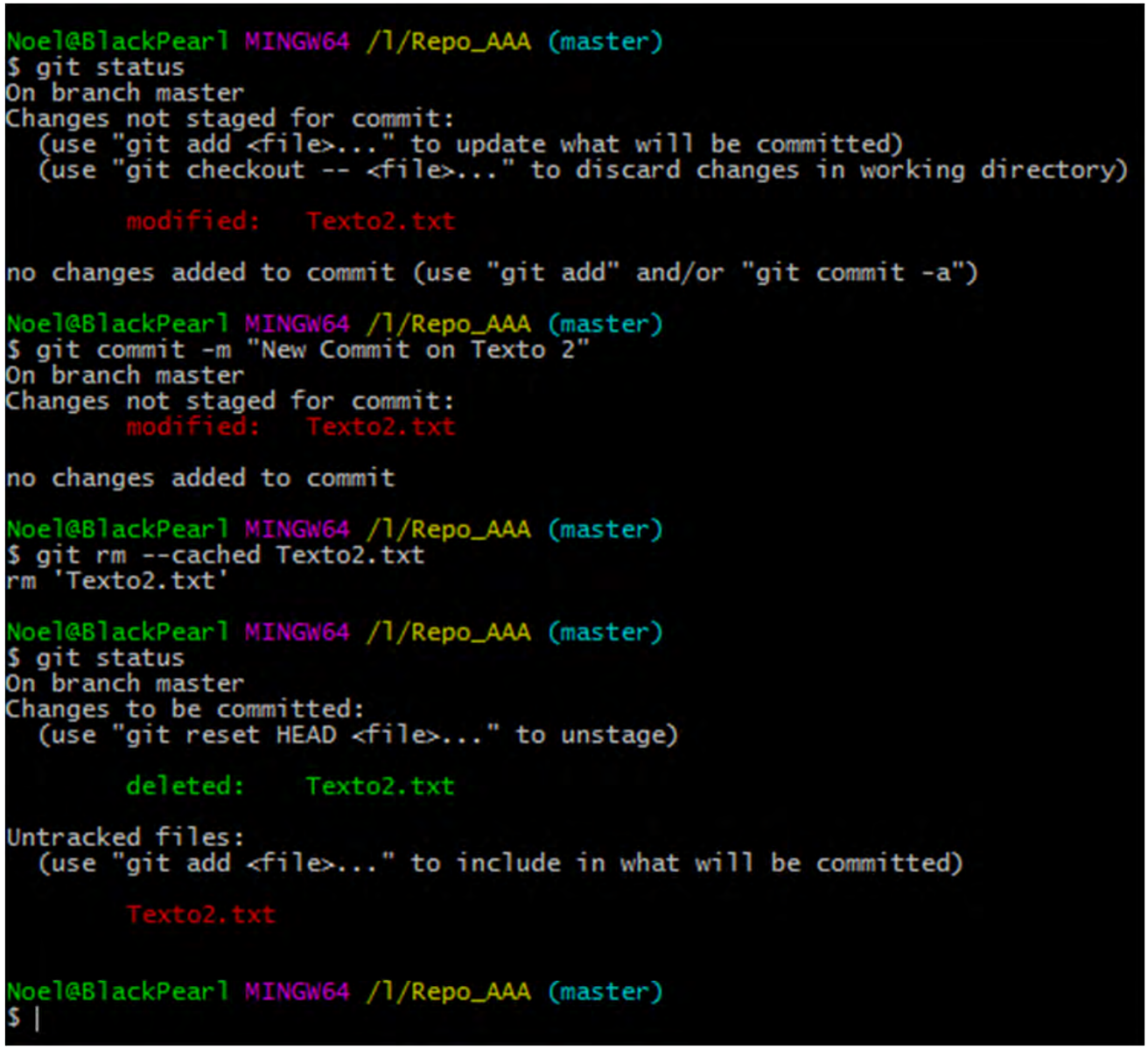
#Before first commit

$ git rm --cached <file>

#After first commit

$ git reset HEAD <files>

GIT – Área de Trabajo



GIT – Historial

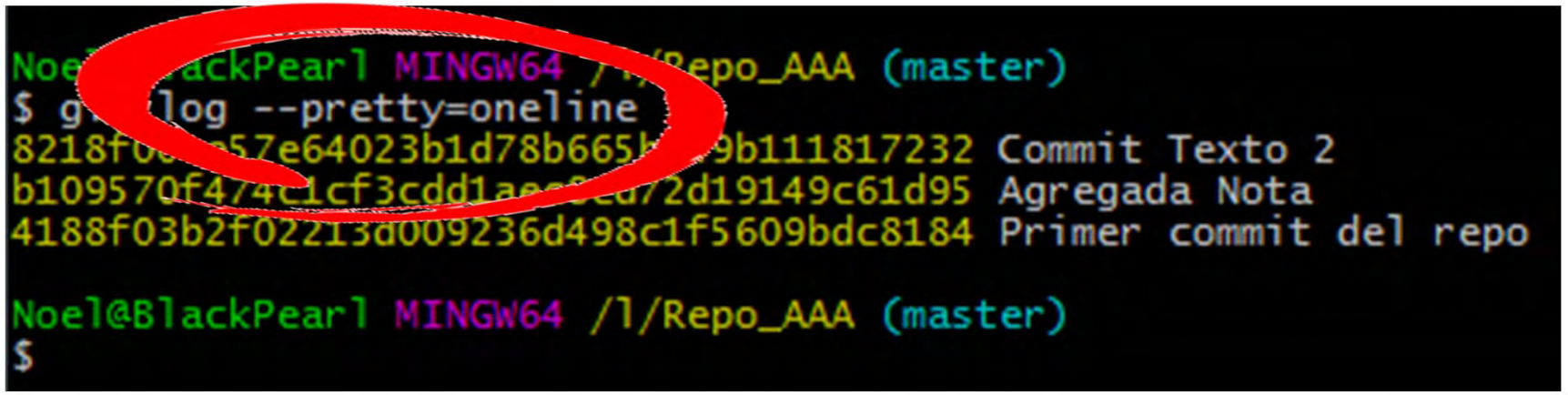
GIT – Historial

NETT FORMACIÓN

```
Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$ git log
commit 8218f00ee57e64023b1d78b665b549b111817232
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 11:22:32 2017 +0100

    Commit Texto 2

commit b109570f474c1cf3cdd1aec8ed72d19149c61d95
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 11:08:21 2017 +0100

    Agregada Nota

commit 4188f03b2f02213d009236d498c1f5609bdc8184
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 10:05:23 2017 +0100

    Primer commit del repo

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$
```

GIT – Historial

NETT FORMACIÓN



git log --pretty=oneline --max-count=2

git log --pretty=oneline --since='5 minutes ago'

git log --pretty=oneline --until='5 minutes ago'

git log --pretty=oneline --author=

git log --pretty=oneline --all

GIT – Historial

GIT – Historial

NETT FORMACIÓN

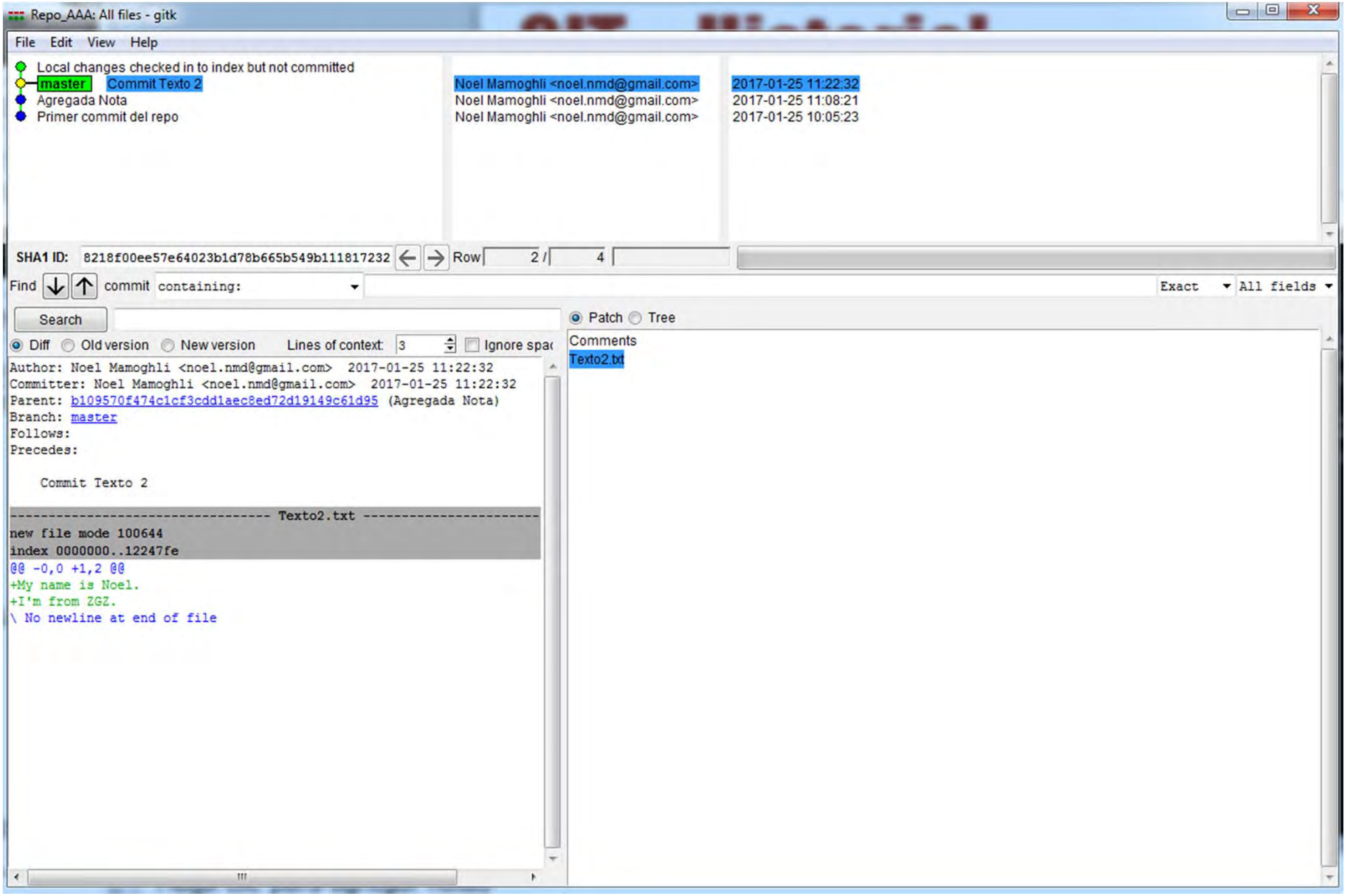
```
Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$ git log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short
* 8218f00 2017-01-25 | Commit Texto 2 (HEAD -> master) [Noel Mamoghli]
* b109570 2017-01-25 | Agregada Nota [Noel Mamoghli]
* 4188f03 2017-01-25 | Primer commit del repo [Noel Mamoghli]

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$
```

- `--pretty="..."` defines the output format.
- `%h` is the abbreviated hash of the commit
- `%d` commit decorations (e.g. branch heads or tags)
- `%ad` is the commit date
- `%s` is the comment
- `%an` is the name of the author
- `--graph` tells git to display the commit tree in the form of an ASCII graph layout
- `--date=short` keeps the date format short and nice

GIT – Historial

NETT FORMACIÓN

GIT – Historial



GIT - Configuración


```
git config --global alias.co checkout
git config --global alias.ci commit
git config --global alias.st status
git config --global alias.br branch
git config --global alias.hist "log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short"
git config --global alias.type 'cat-file -t'
git config --global alias.dump 'cat-file -p'
```

GIT - Configuración

NETT FORMACIÓN

git config --global alias.co checkout
git config --global alias.ci commit

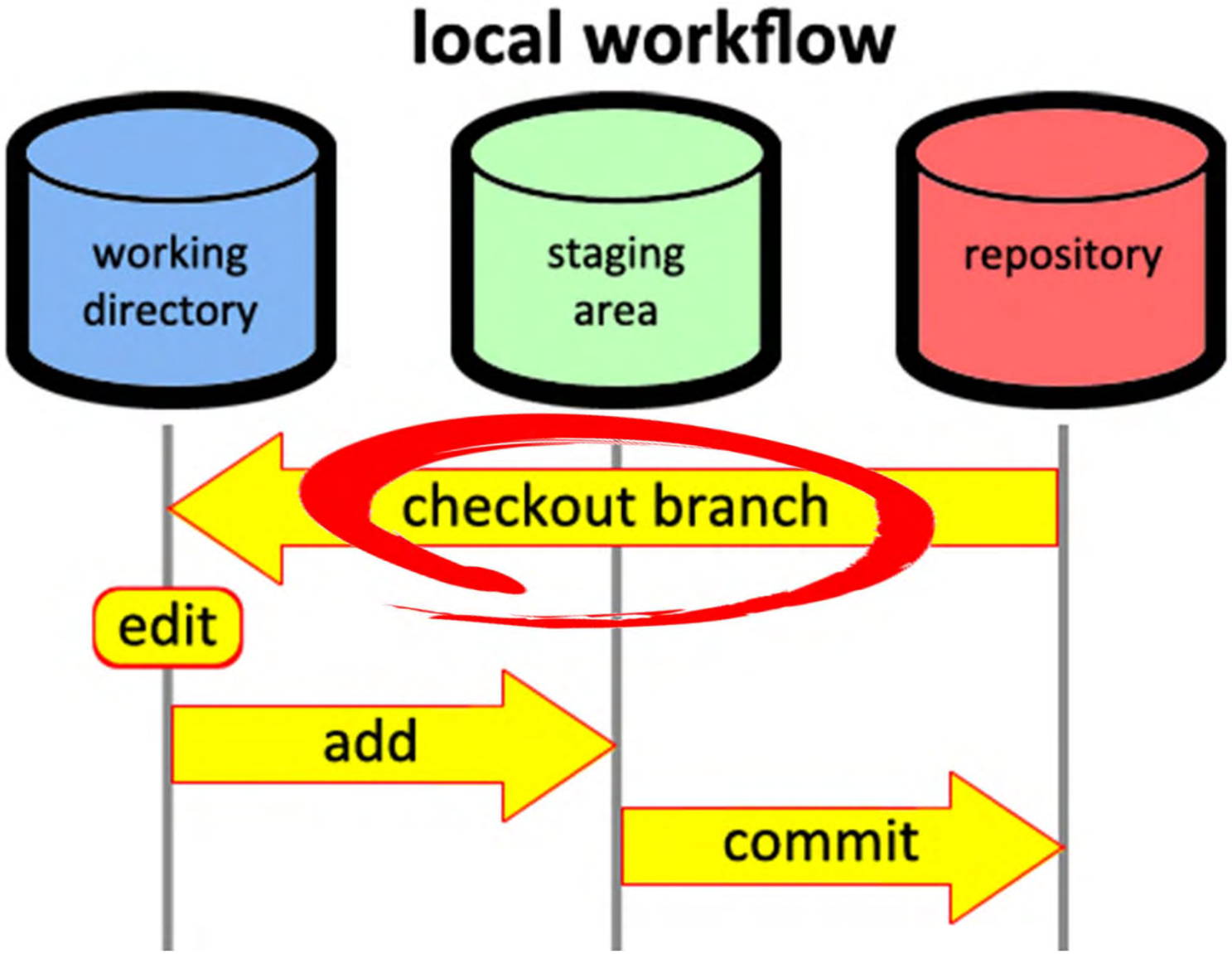
FILE: *.GITCONFIG*

```
[alias]
  co = checkout
  ci = commit
  st = status
  br = branch
  hist = log --pretty=format:\"%h %ad | %s%d [%an]\" --graph --date=short
  type = cat-file -t
  dump = cat-file -p
```

GIT – Área de Trabajo

NETT FORMACIÓN

GIT – Área de Trabajo



NETT FORMACIÓN

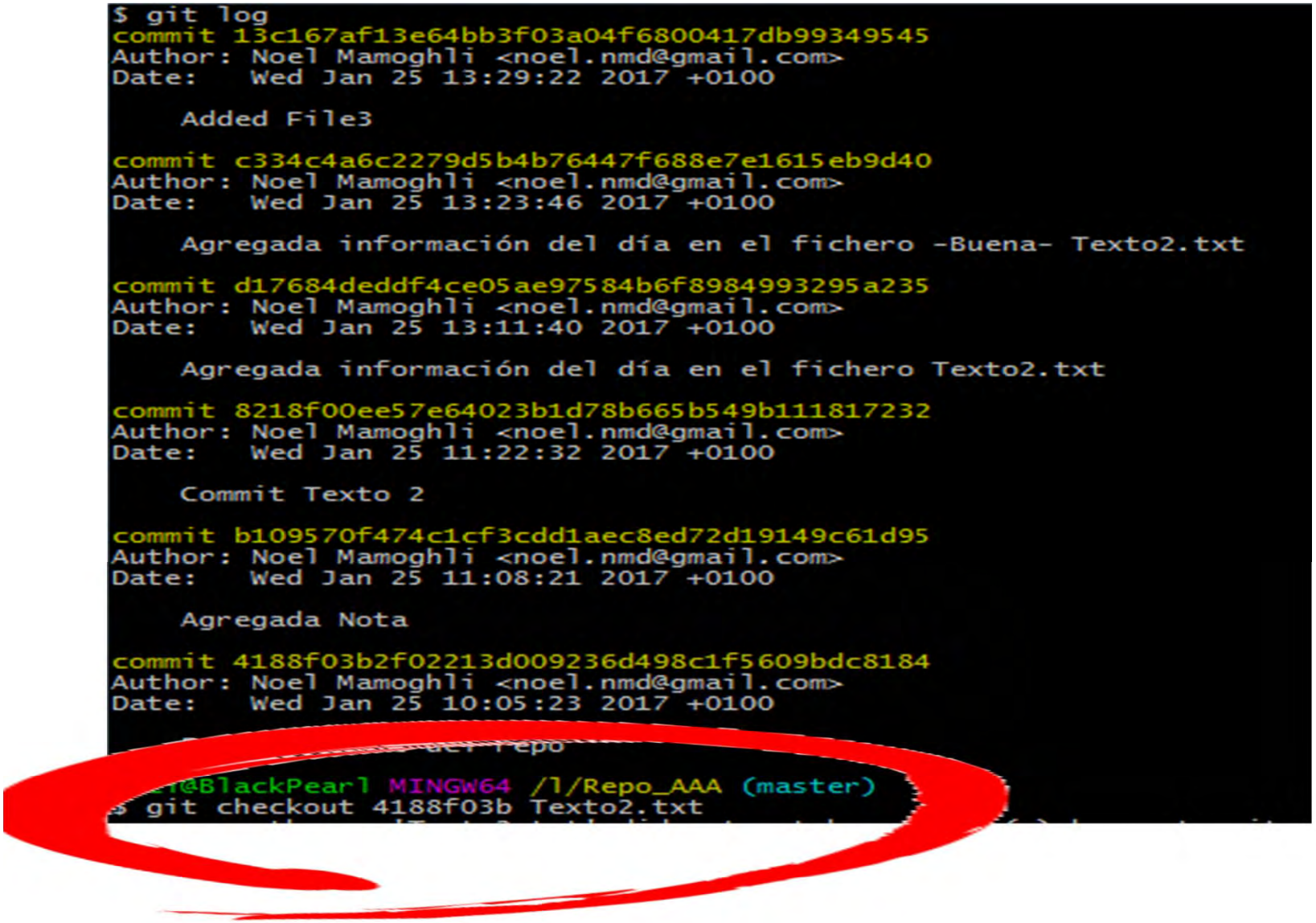
1. Discard changes in a file in the index to the copy in working directory

```
$ git checkout -- <files>
```

2. Reverting a file some commits before

```
$ git checkout <commit-id> file
```

GIT – Área de Trabajo

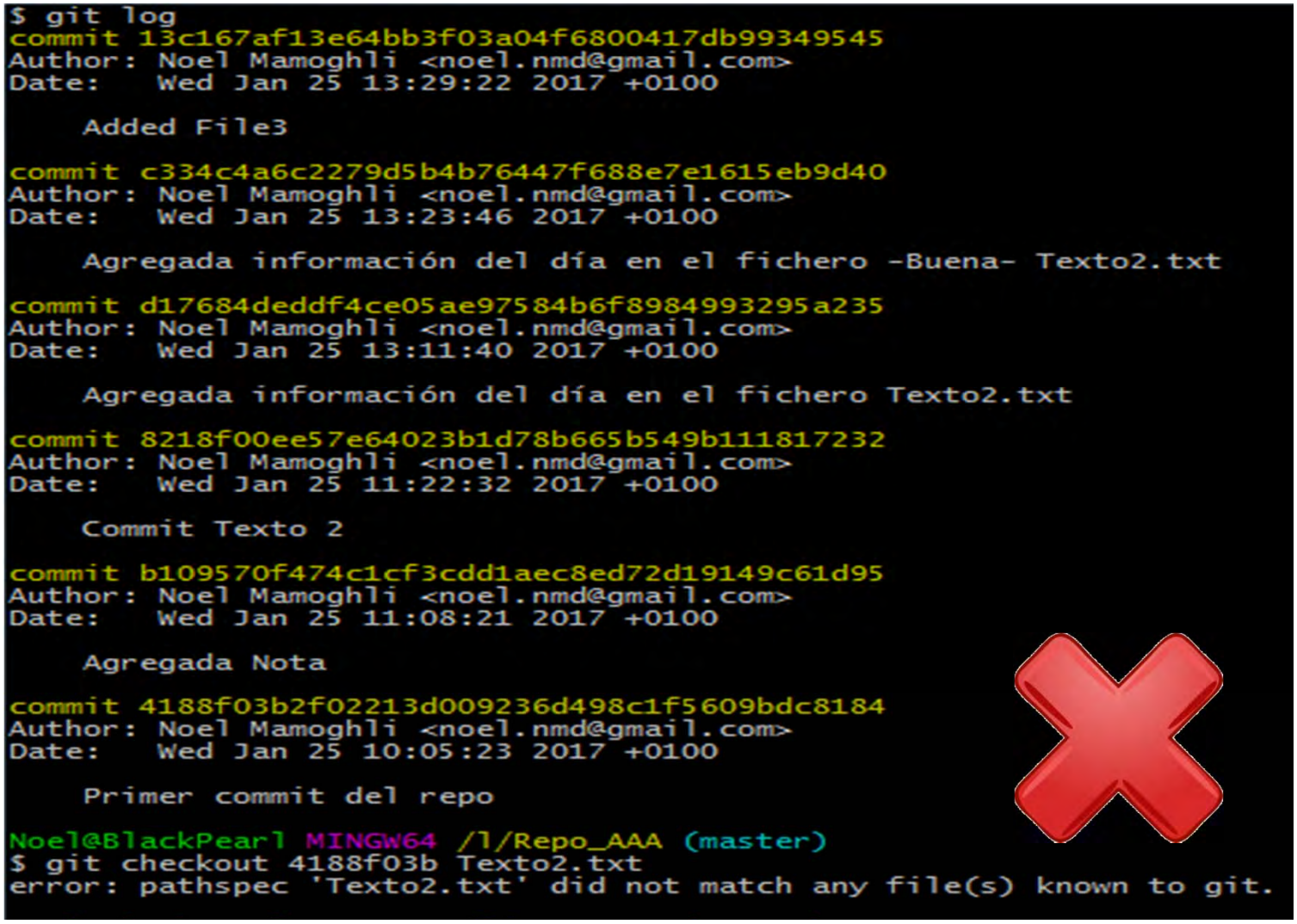


GIT – Área de Trabajo

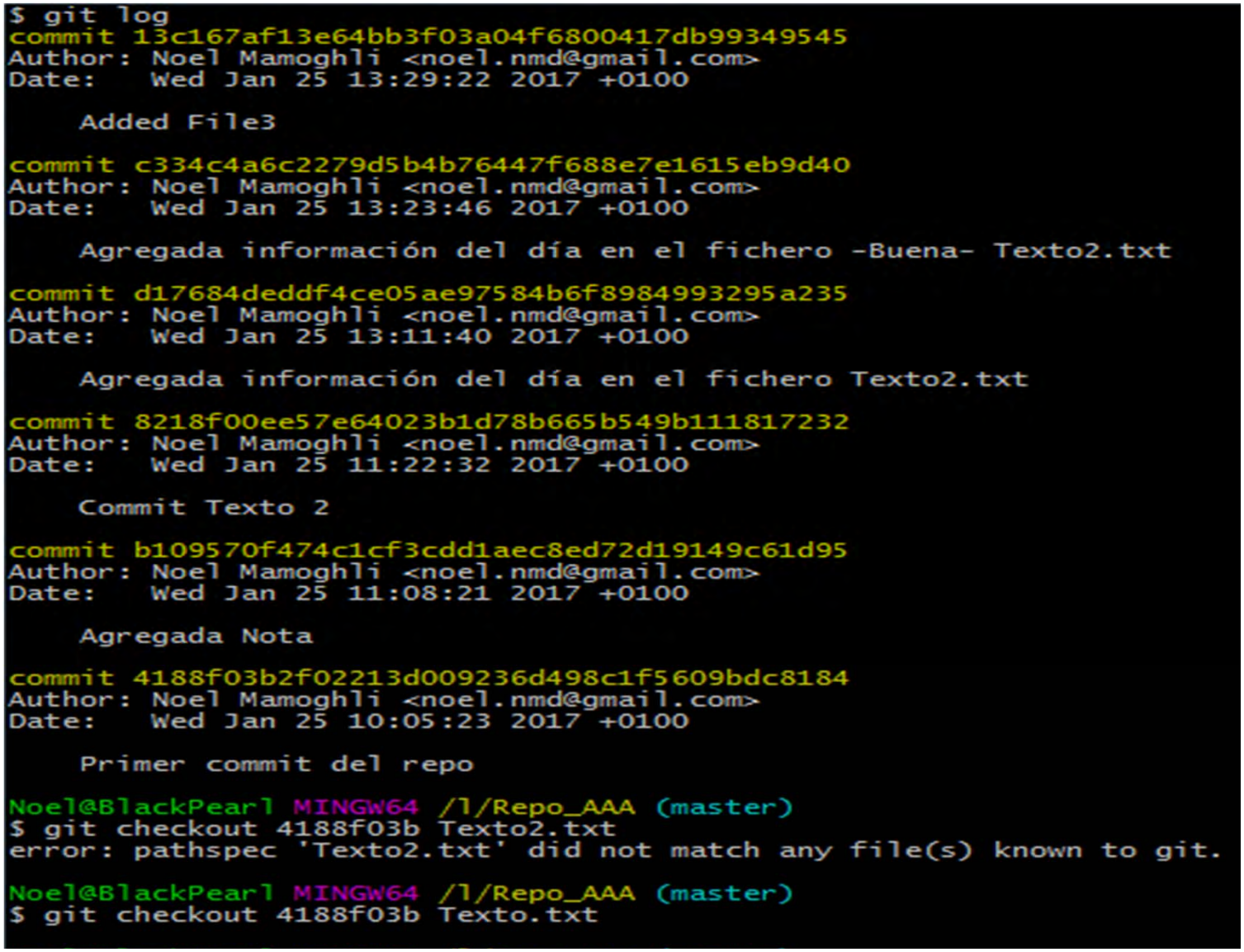
NETT FORMACIÓN

```
$ git log
commit 13c167af13e64bb3f03a04f6800417db99349545
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 13:29:22 2017 +0100

    Added File3

commit c334c4a6c2279d5b4b76447f688e7e1615eb9d40
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 13:23:46 2017 +0100

    Agregada información del día en el fichero -Buena- Texto2.txt

commit d17684deddf4ce05ae97584b6f8984993295a235
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 13:11:40 2017 +0100

    Agregada información del día en el fichero Texto2.txt

commit 8218f00ee57e64023b1d78b665b549b111817232
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 11:22:32 2017 +0100

    Commit Texto 2

commit b109570f474c1cf3cdd1aec8ed72d19149c61d95
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 11:08:21 2017 +0100

    Agregada Nota

commit 4188f03b2f02213d009236d498c1f5609bdc8184
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 10:05:23 2017 +0100

                    el Repo

el@BlackPearl MINGW64 /l/Repo_AAA (master)
$ git checkout 4188f03b Texto2.txt
```

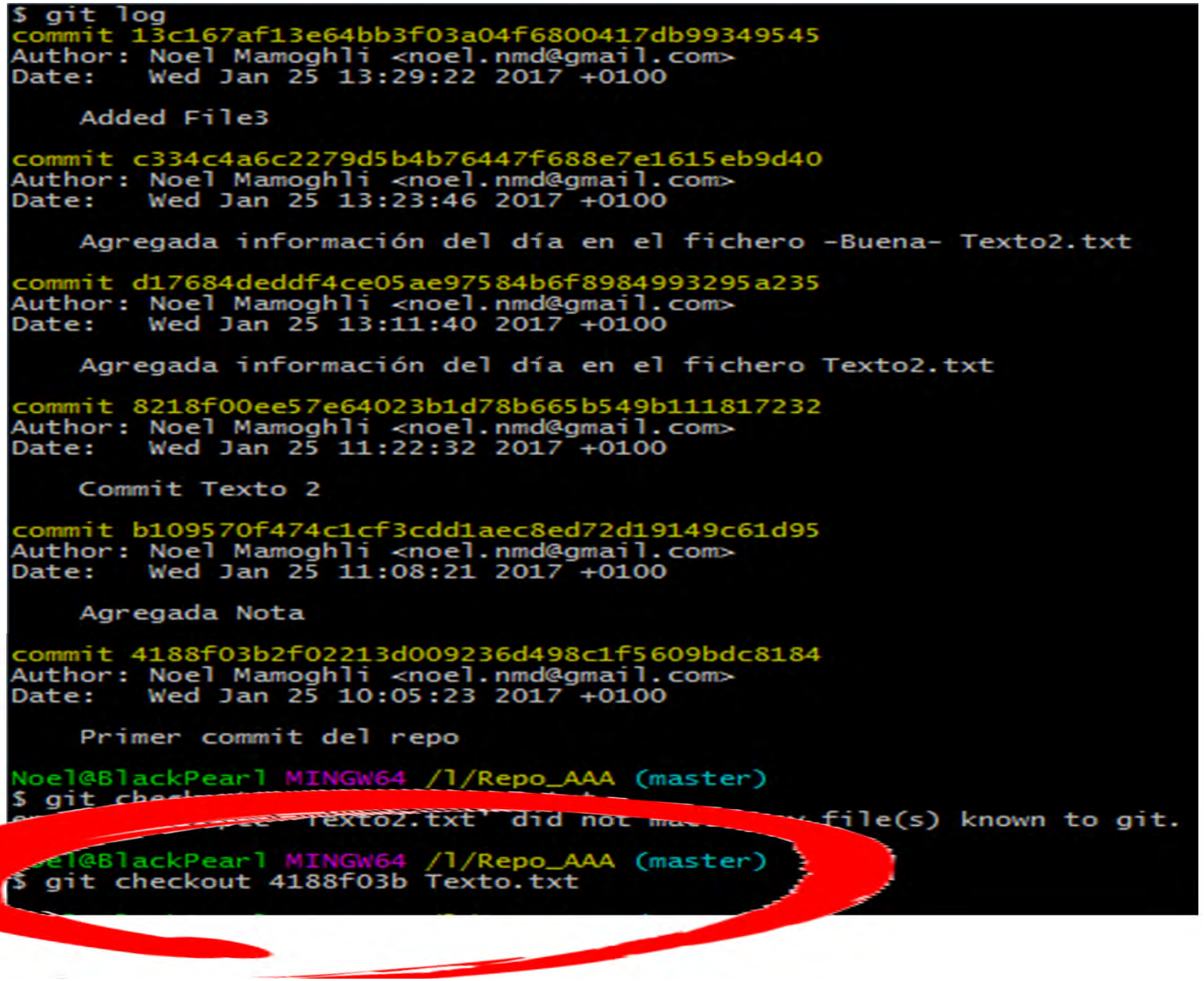
GIT – Área de Trabajo



GIT – Área de Trabajo



GIT – Área de Trabajo



GIT – Área de Trabajo

NETT FORMACIÓN

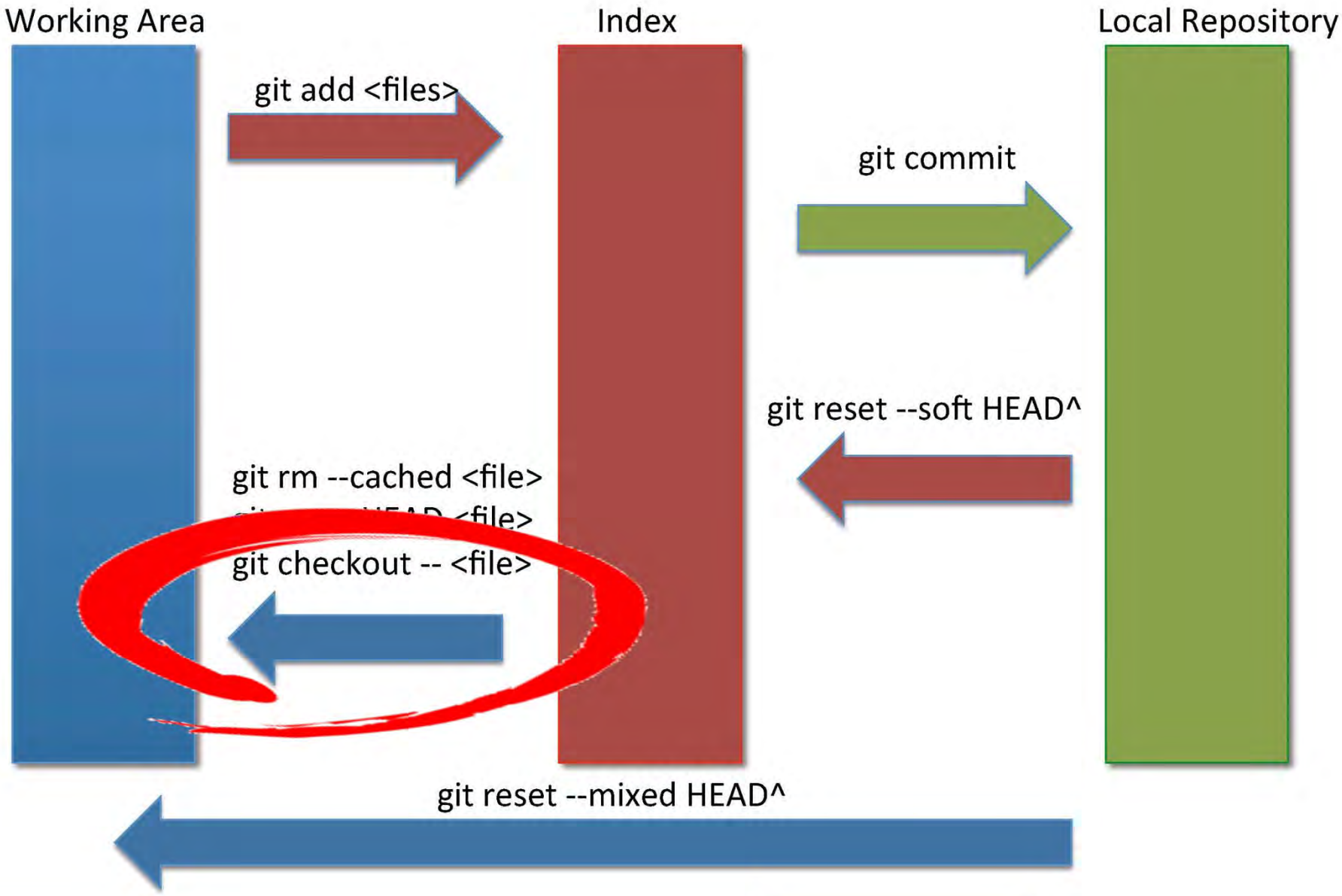
```
$ git log
commit 13c167af13e64bb3f03a04f6800417db99349545
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 13:29:22 2017 +0100

    Added File3

commit c334c4a6c2279d5b4b76447f688e7e1615eb9d40
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 13:23:46 2017 +0100

    Agregada información del día en el fichero -Buena- Texto2.txt

commit d17684deddf4ce05ae97584b6f8984993295a235
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 13:11:40 2017 +0100

    Agregada información del día en el fichero Texto2.txt

commit 8218f00ee57e64023b1d78b665b549b111817232
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 11:22:32 2017 +0100

    Commit Texto 2

commit b109570f474c1cf3cdd1aec8ed72d19149c61d95
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 11:08:21 2017 +0100

    Agregada Nota

commit 4188f03b2f02213d009236d498c1f5609bdc8184
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 10:05:23 2017 +0100

    Primer commit del repo

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$ git ch
        Texto2.txt' did not
                                file(s) know

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$ git checkout 4188f03b Texto.txt
```

GIT – Área de Trabajo

NETT FORMACIÓN

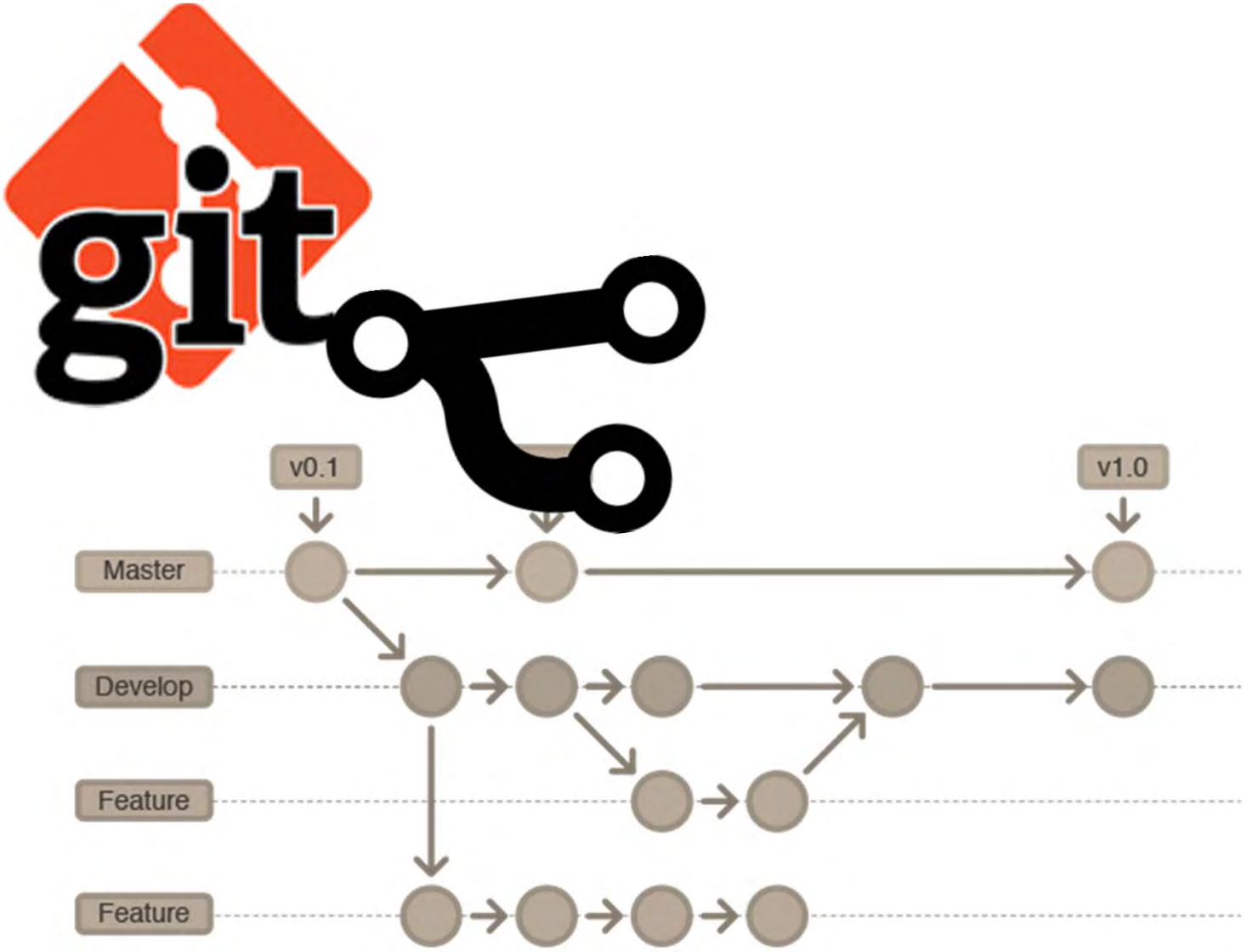
```
$ git log
commit 13c167af13e64bb3f03a04f6800417db99349545
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 13:29:22 2017 +0100

    Added File3

commit c334c4a6c2279d5b4b76447f688e7e1615eb9d40
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 13:23:46 2017 +0100

    Agregada información del día en el fichero -Buena- Texto2.txt

commit d17684deddf4ce05ae97584b6f8984993295a235
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 13:11:40 2017 +0100

    Agregada información del día en el fichero Texto2.txt

commit 8218f00ee57e64023b1d78b665b549b111817232
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 11:22:32 2017 +0100

    Commit Texto 2

commit b109570f474c1cf3cdd1aec8ed72d19149c61d95
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 11:08:21 2017 +0100

    Agregada Nota

commit 4188f03b2f02213d009236d498c1f5609bdc8184
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 10:05:23 2017 +0100

    Primer commit del repo

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$ git checkout 4188f03b Texto2.txt
error: pathspec 'Texto2.txt' did not match any file(s) known to git.

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$ git checkout 4188f03b Texto.txt

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$
```

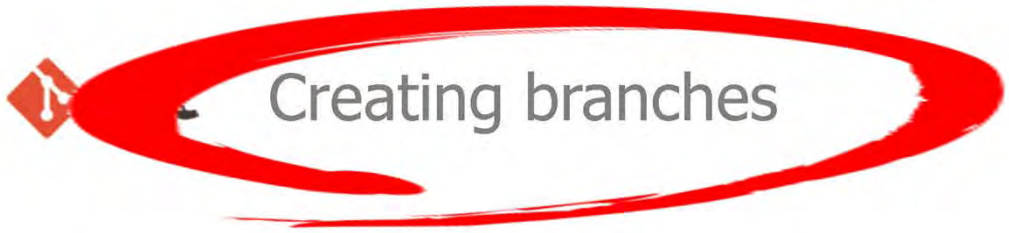
Basic Flow: Reset



GIT – Ramas

NETT FORMACIÓN





Creating branches

NETT FORMACIÓN

From the active branch:

$ git branch <branch-name>: Only creates a branch

$ git checkout -b <branch-name>: Creates and switches to the new branch

Listing branches

List branches names found in the repository:

$ git branch

```
  bug/pr-1
  dev
* master
```

$ git branch –a

Lists all branches (also tracked branches)

git Viewing branches

More detailed output than git branch.

Listing the commits that contribute to one or more branches

$ git show-branch

```
! [bug/pr-1] Fix Problem Report 1
 * [dev] Improve the new development
  ! [master] Added Bob's fixes.
---
 *  [dev] Improve the new development
 *  [dev^] Start some new development.
+   [bug/pr-1] Fix Problem Report 1
+*+ [master] Added Bob's fixes.
```

-r shows remote tracking branches

-a shows all branches

Switching branches

Your working directory can reflect only one branch at a time.

To start working on a different branch:

$ git checkout <branch-name>

git Delete local branches

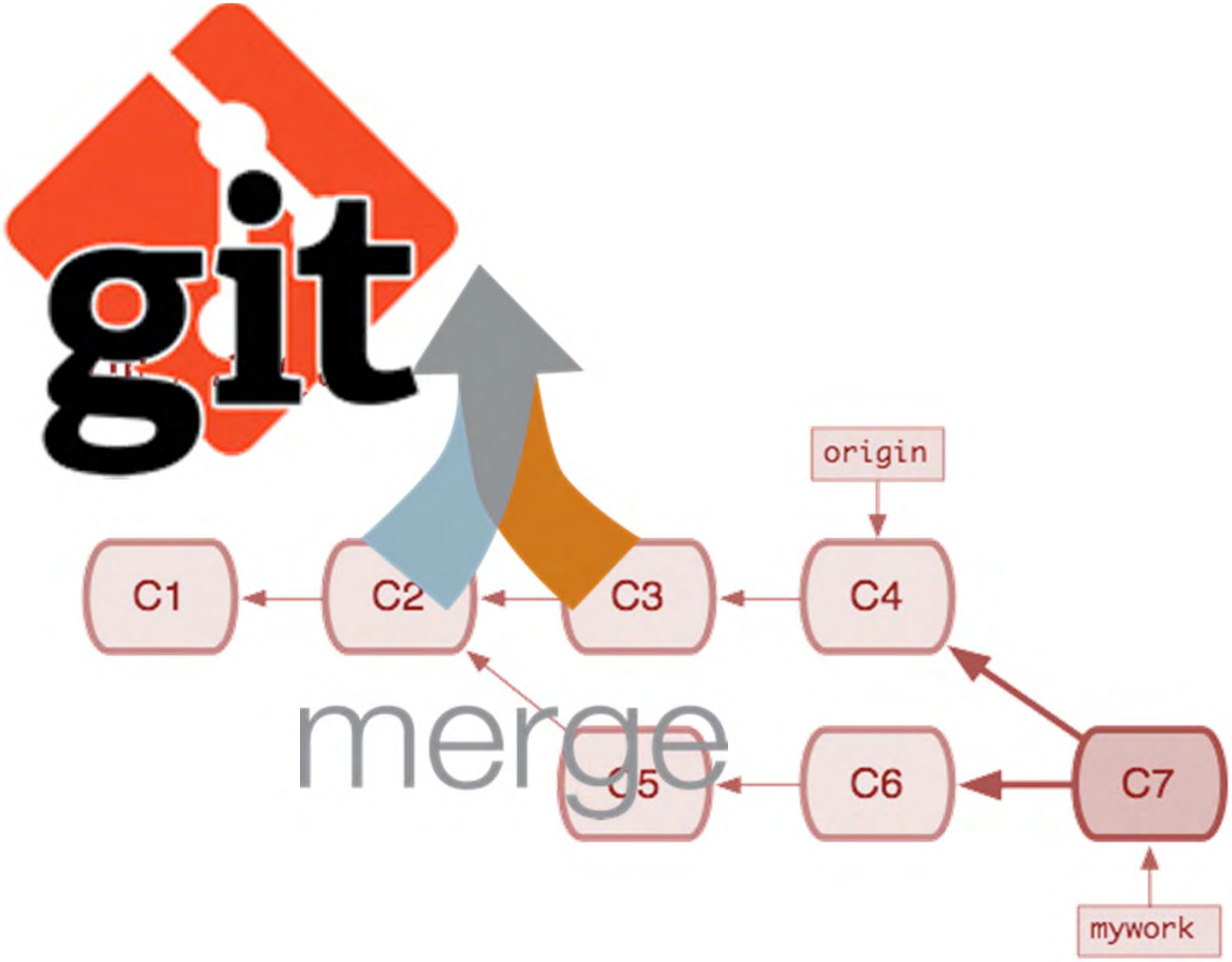
NETT FORMACIÓN

You cannot delete the active branch.

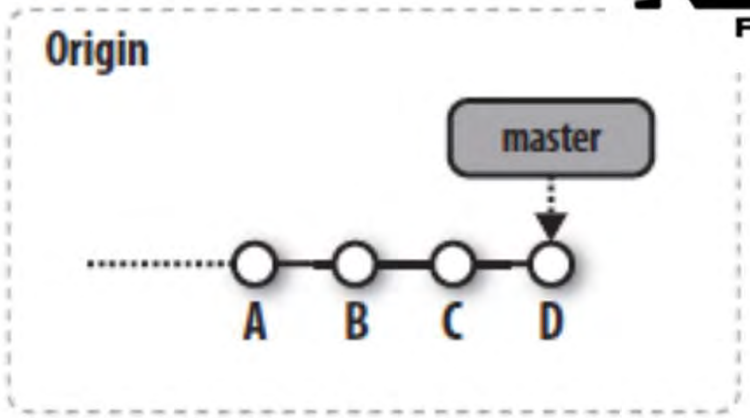
$ git branch –d <branch>: Deletes if you have done merge.

$ git branch –D <branch> : Force the action even though you haven't done merge.

GIT – Ramas

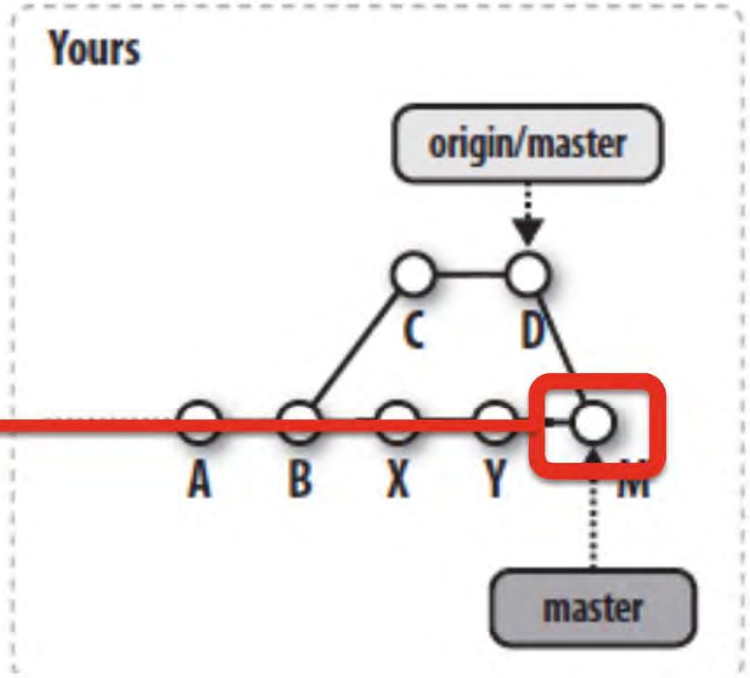


git Merge



$ git merge <branch>

New commit is created



git What is a merge

NETT FORMACIÓN

When modifications in one branch do not conflict in another branch → new commit

When branches conflict (alter the same line) Git does not resolve the dispute.

git Merge examples

NETT FORMACIÓN

You have to switch to the target branch and execute the merge:

```
$ git checkout destinyBranch
```

```
$ git merge anotherBranch
```

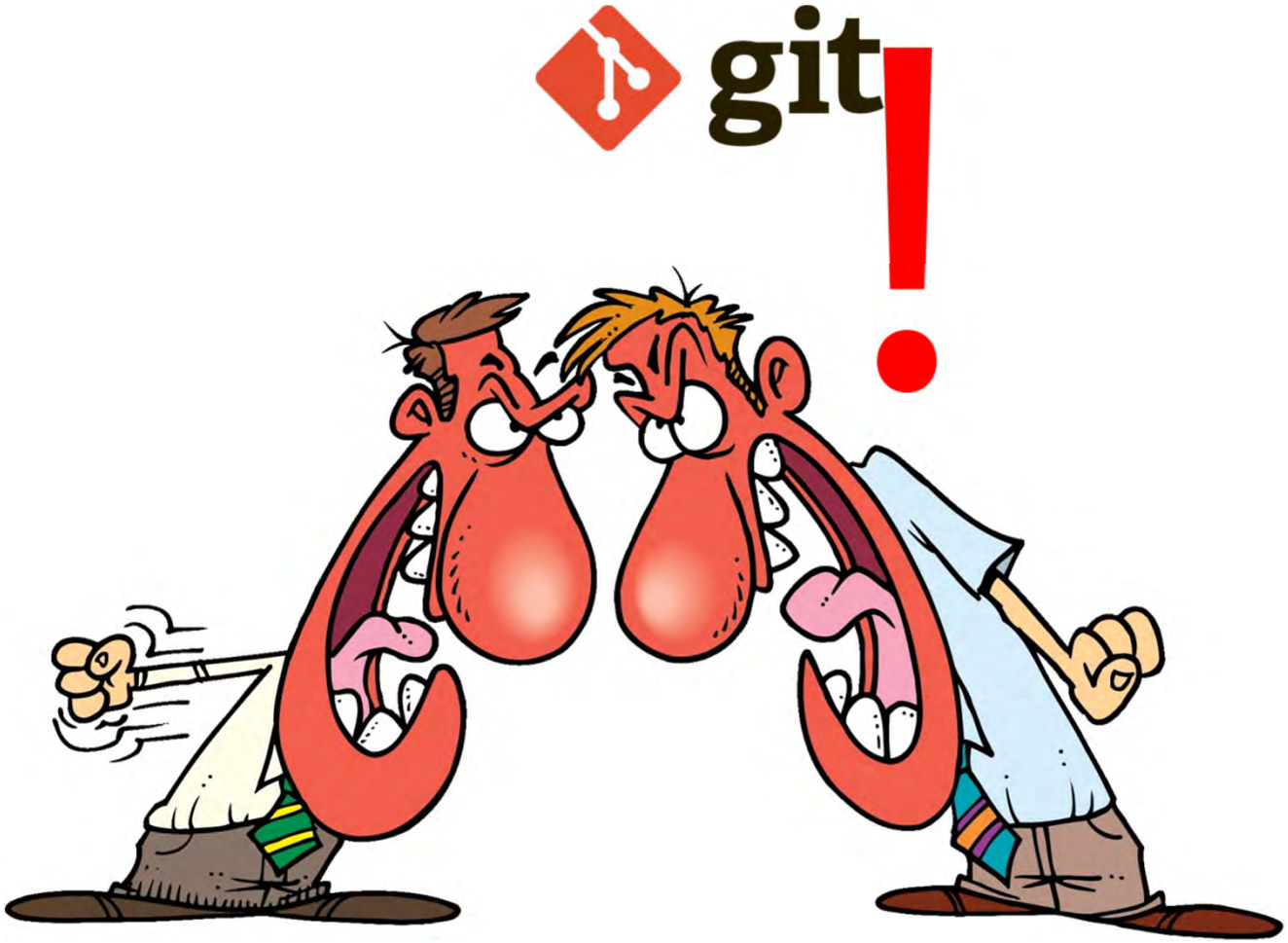
NETT FORMACIÓN

First of begin a merge → tidy up working directory

If you start a merge in a dirty state, Git may be unable to combine the changes from all the branches and those in your working directory or index in one pass.

git

NETT FORMACIÓN



git Merge Conflicts

Git warns you about the conflict:

```
israelalcazar@Mac-Lamaro:~/dev/git/examples2 (master) $ git merge newFeature
Auto-merging README3.TXT
CONFLICT (content): Merge conflict in README3.TXT
Automatic merge failed; fix conflicts and then commit the result.
```

And marks the file:

git Locating conflicted Files

NETT FORMACIÓN

Which files have got conflicts?

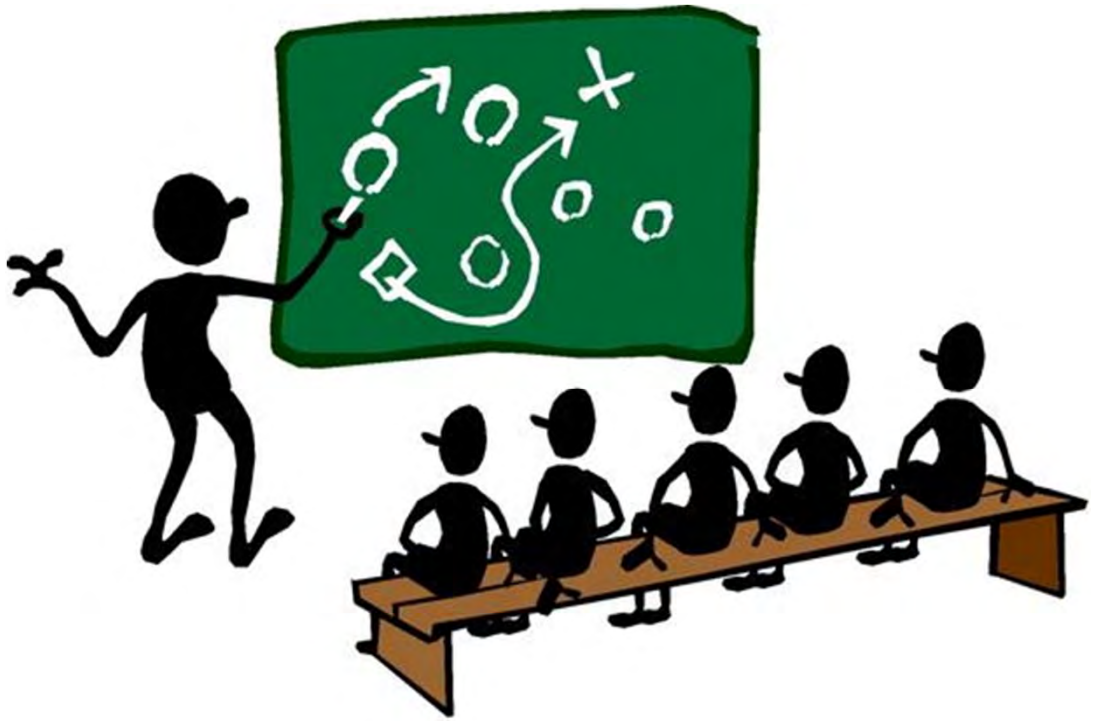
Git keeps track of problematic files by marking each one in the index as conflicted or unmerged

You can use one of these commands:

$ git status

$ git diff

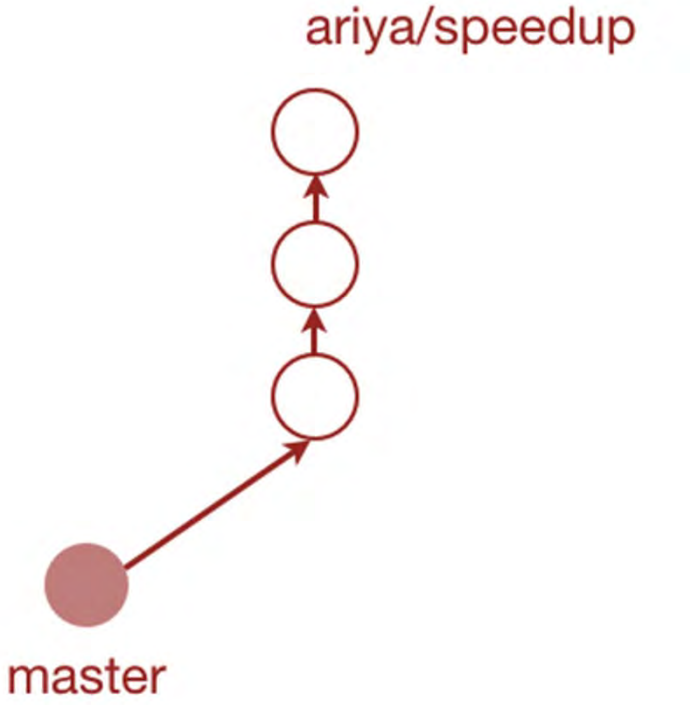
Fast-Forward
vs Non Fast-Forward



Merge: Fast-Forward vs Non Fast-Forward

Situación Inicial

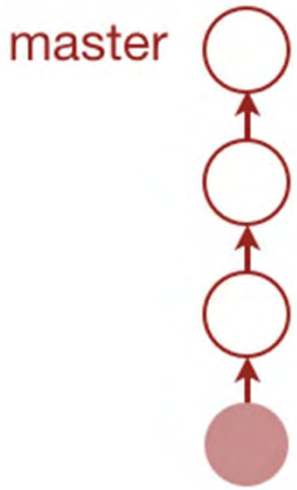
```
git checkout -b speedup
```



Merge: Fast-Forward vs Non Fast-Forward

merge fast -forward (por defecto)

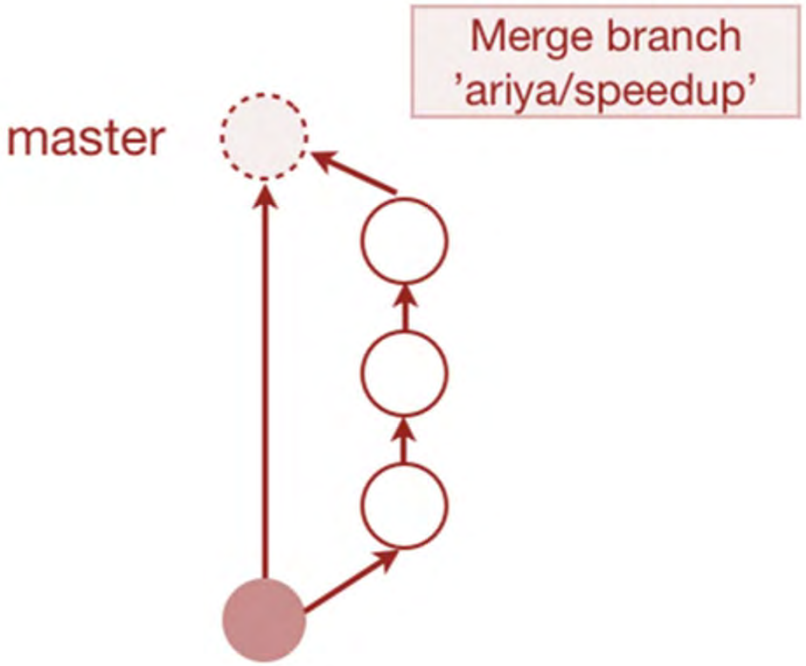
```
git fetch ariya
git merge ariya/speedup
```



Merge: Fast-Forward vs Non Fast-Forward

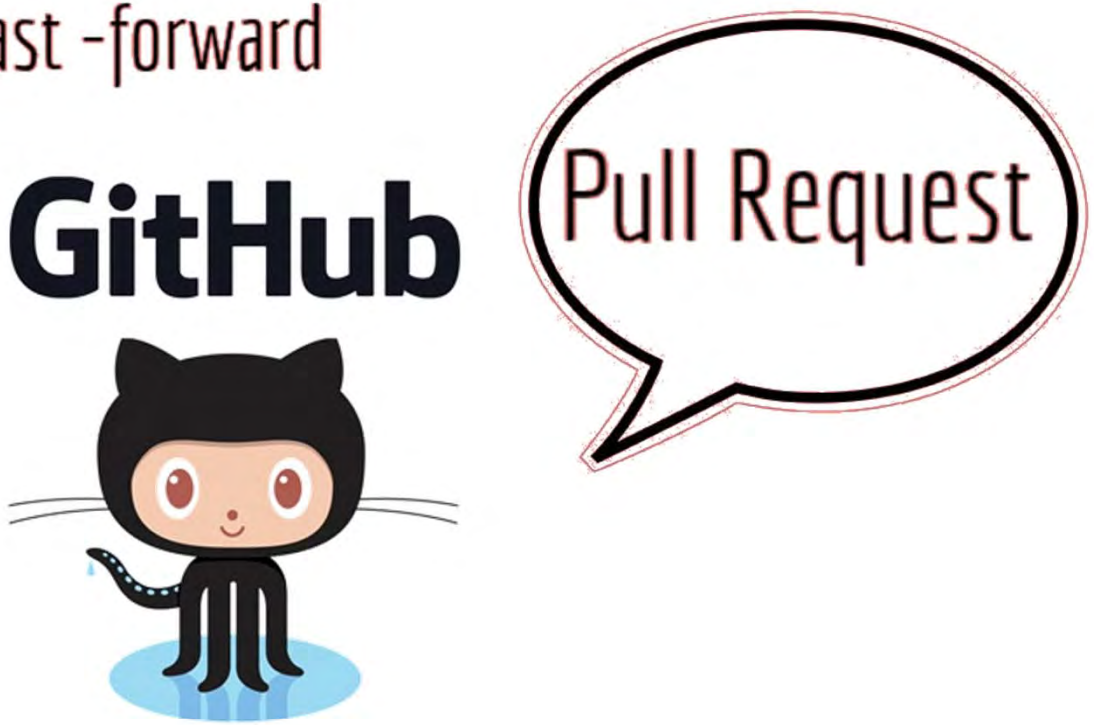
merge non fast -forward

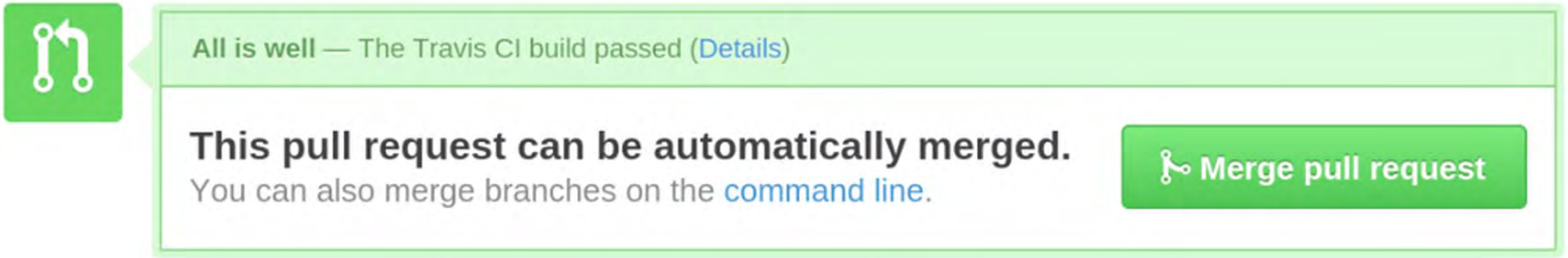
```
git fetch ariya
git merge -no-ff ariya/speedup
```



Merge: Fast-Forward vs Non Fast-Forward

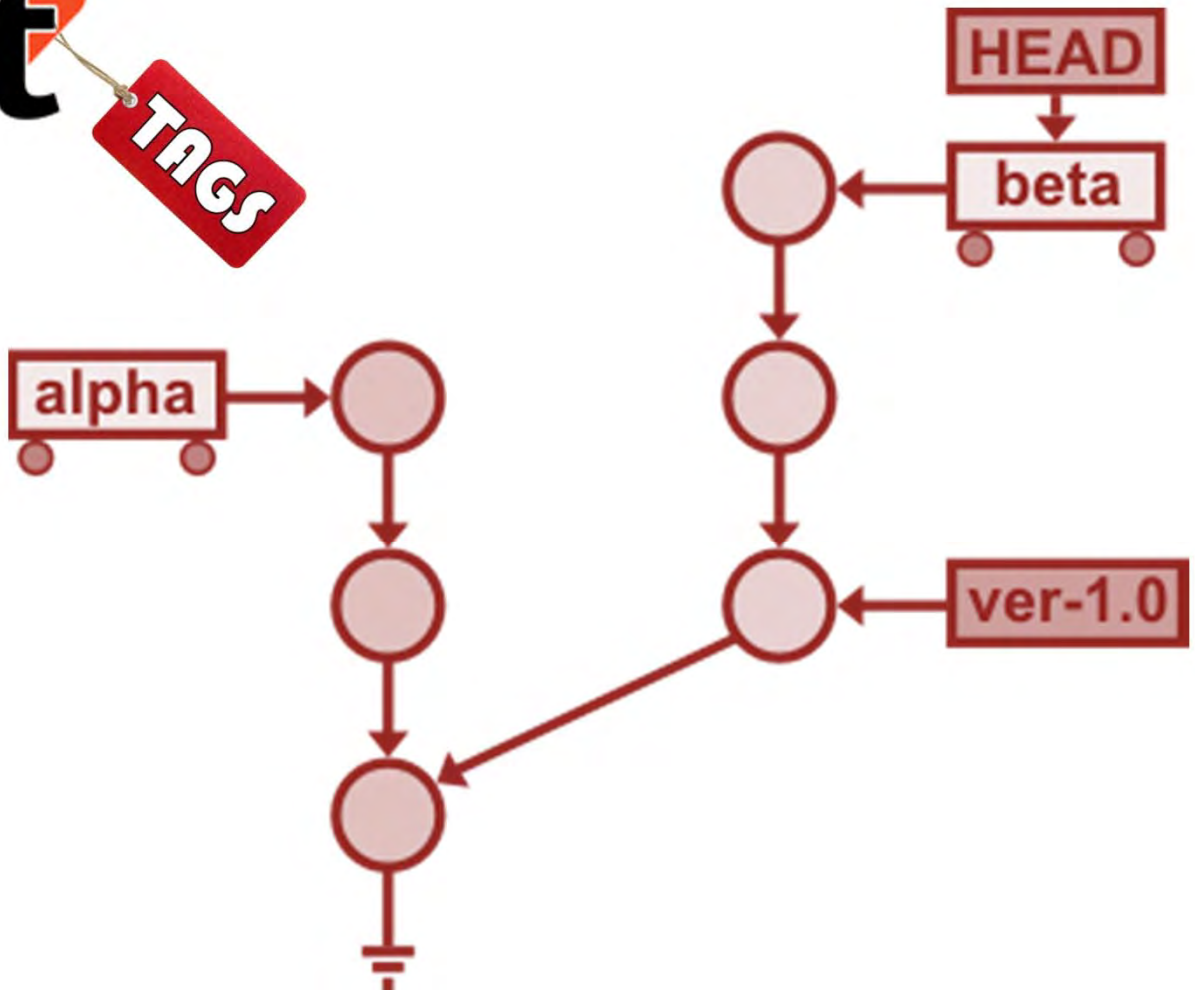
merge non fast -forward





GIT - Tagging

NETT FORMACIÓN



git Tags vs Branches

NETT FORMACIÓN

A tag is meant to be a static name that does not change or move over time. Once applied, you should leave it alone.

A branch is dynamic and moves with each commit you make. The branch name is designed to follow your continuing development.

You can give a branch and a tag the same name.

NETT FORMACIÓN

$ git tag : List tags

$ git tag –a <name> -m “text to …”: Create a tag

$ git show <name>

git Commands

$ git push <repo> --tags: Sends all tags

$ git push <repo> <tag-name>: Sends the tag

NETT FORMACIÓN

- **Lightweight**

Like a branch that doesn't change. A pointer to a specific commit

- **Annotated**

Are stored as full objects in the Git database. They are checksummed; contain the tagger name, email and date. Can be signed and verified.

GIT - Tagging

Agregar Tags

```
Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$ git tag -a v0.3 -m "Version 0.3"

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$ git tag
v0.3

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$ git show v0.3
tag v0.3
Tagger: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 14:15:47 2017 +0100

Version 0.3

commit 13c167af13e64bb3f03a04f6800417db99349545
Author: Noel Mamoghli <noel.nmd@gmail.com>
Date:   Wed Jan 25 13:29:22 2017 +0100

    Added File3

diff --git a/Texto3.txt b/Texto3.txt
new file mode 100644
index 0000000..e75cd02
--- /dev/null
+++ b/Texto3.txt
@@ -0,0 +1,5 @@
+<h1>Hold on to your butts</h1>^M
+<p>My money's in that office, right? If she start giving me some bullshit about it ain't there, and we got to go someplace else and get it.^M
+I'm gonna shoot you in the head then and there.^M
+Then I'm gonna shoot that bitch in the kneecaps, find out where my goddamn money is. She gonna tell me too. Hey, look at me when I'm talking to y
ou, motherfucker.^M
+You listen: we go in there, and that nigga Winston or anybody else is in there, you the first motherfucker to get shot. You understand? </p>^M

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$
```

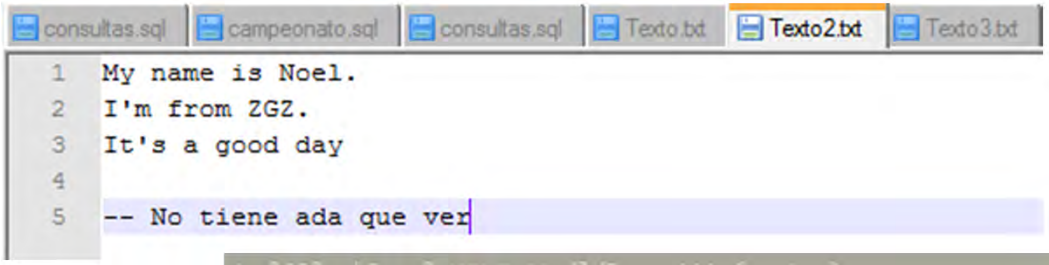
GIT - Tagging

Eliminar Tags

```
mamoghli@MAMOGHLI2 MINGW64 /c/HR-Personal/AAA/Repo_AAA (master)
$ git tag
v0.0.01_SNAPSHOT

mamoghli@MAMOGHLI2 MINGW64 /c/HR-Personal/AAA/Repo_AAA (master)
$ git tag -d v0.0.01_SNAPSHOT
Deleted tag 'v0.0.01_SNAPSHOT' (was 9084bf2)

mamoghli@MAMOGHLI2 MINGW64 /c/HR-Personal/AAA/Repo_AAA (master)
$ git tag

mamoghli@MAMOGHLI2 MINGW64 /c/HR-Personal/AAA/Repo_AAA (master)
$
```

GIT – Área de Trabajo

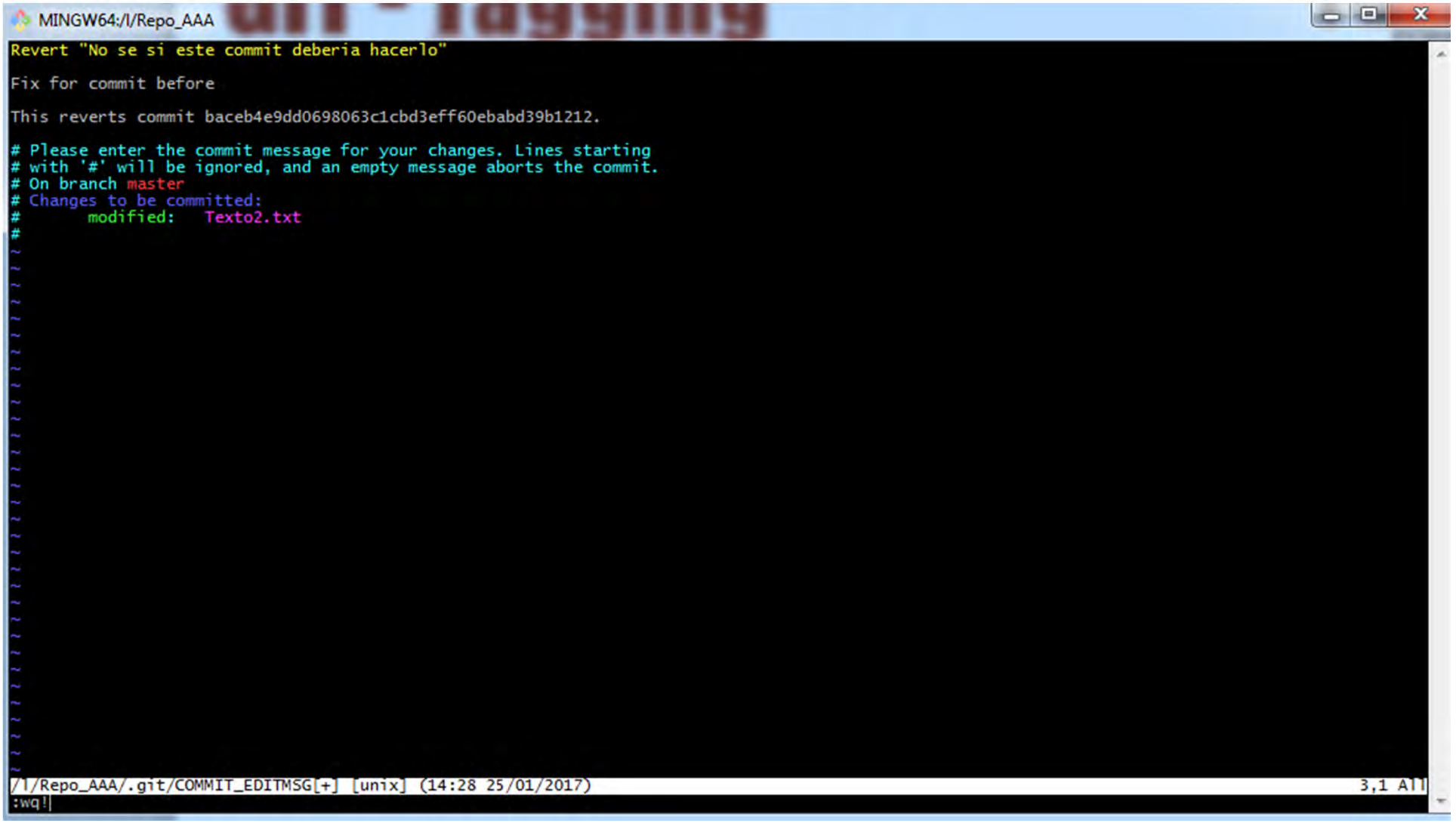
GIT – Área de Trabajo



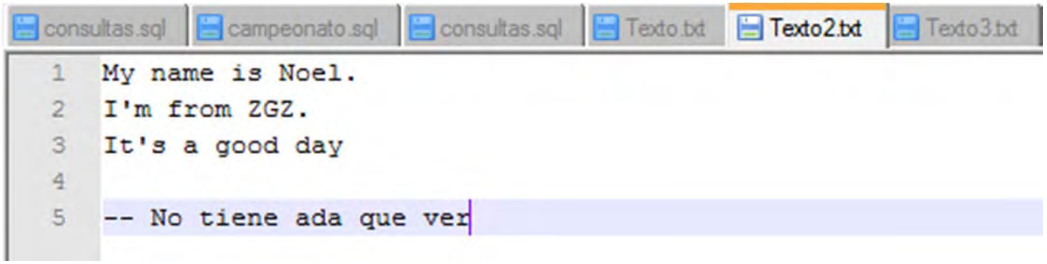

```
Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

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

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$ git commit -am "No se si este commit deberia hacerlo"
[master baceb4e] No se si este commit deberia hacerlo
 1 file changed, 3 insertions(+), 1 deletion(-)

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$ git status
On branch master
nothing to commit, working tree clean

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$
```

```
Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$ git revert HEAD
[master 4b72fc9] Revert "No se si este commit deberia hacerlo"
 1 file changed, 1 insertion(+), 3 deletions(-)

Noel@BlackPearl MINGW64 /l/Repo_AAA (master)
$
```

GIT – Área de Trabajo



GIT – Área de Trabajo



GIT – Área de Trabajo

git Symrefs

NETT FORMACIÓN

HEAD

ORIG_HEAD

FETCH_HEAD

MERGE_HEAD

git Symrefs

NETT FORMACIÓN

^, un commit atrás

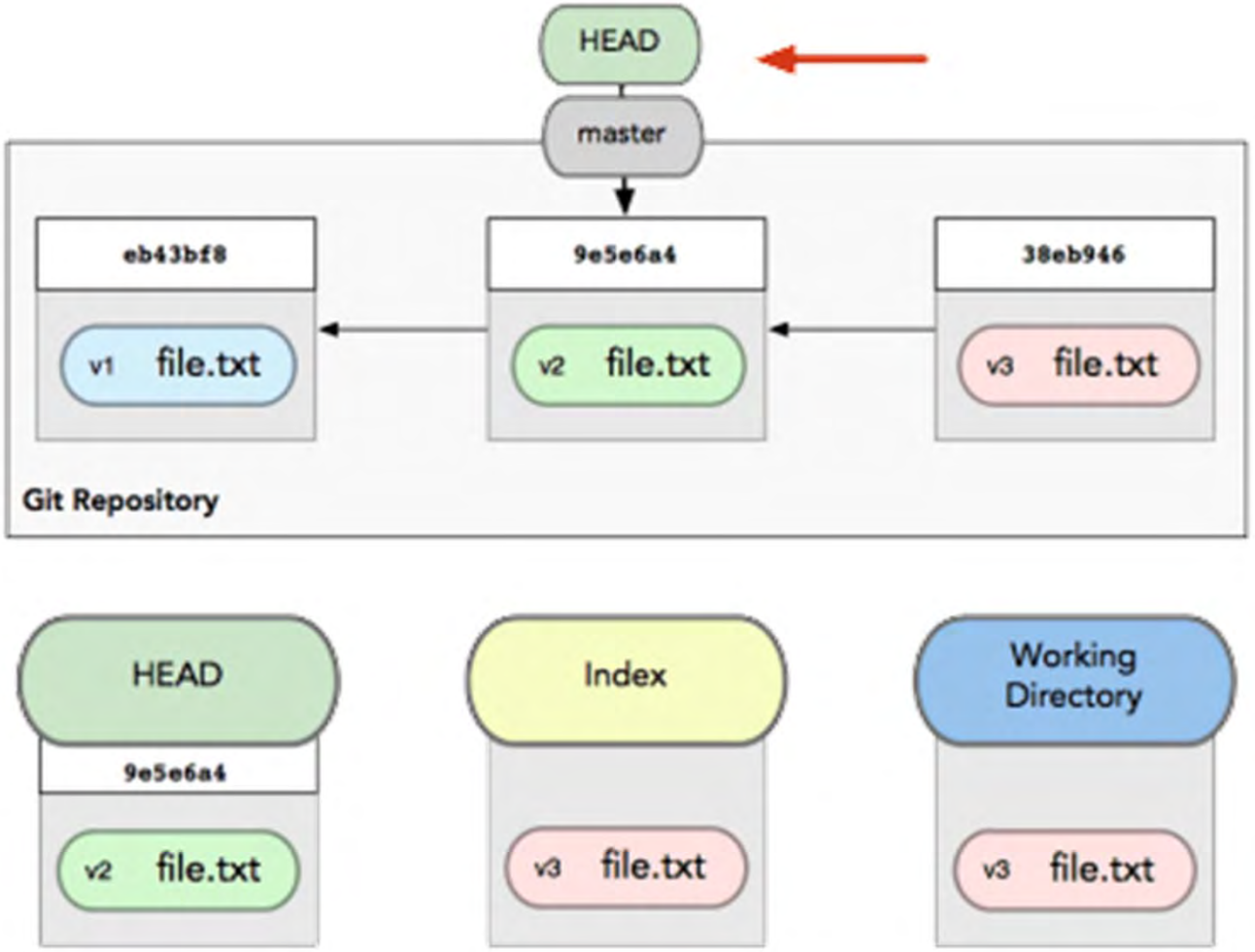
^^^^, 4 comimts atrás (tantos como circunflejos)

~ , un commit atrás

~n, n commits atrás

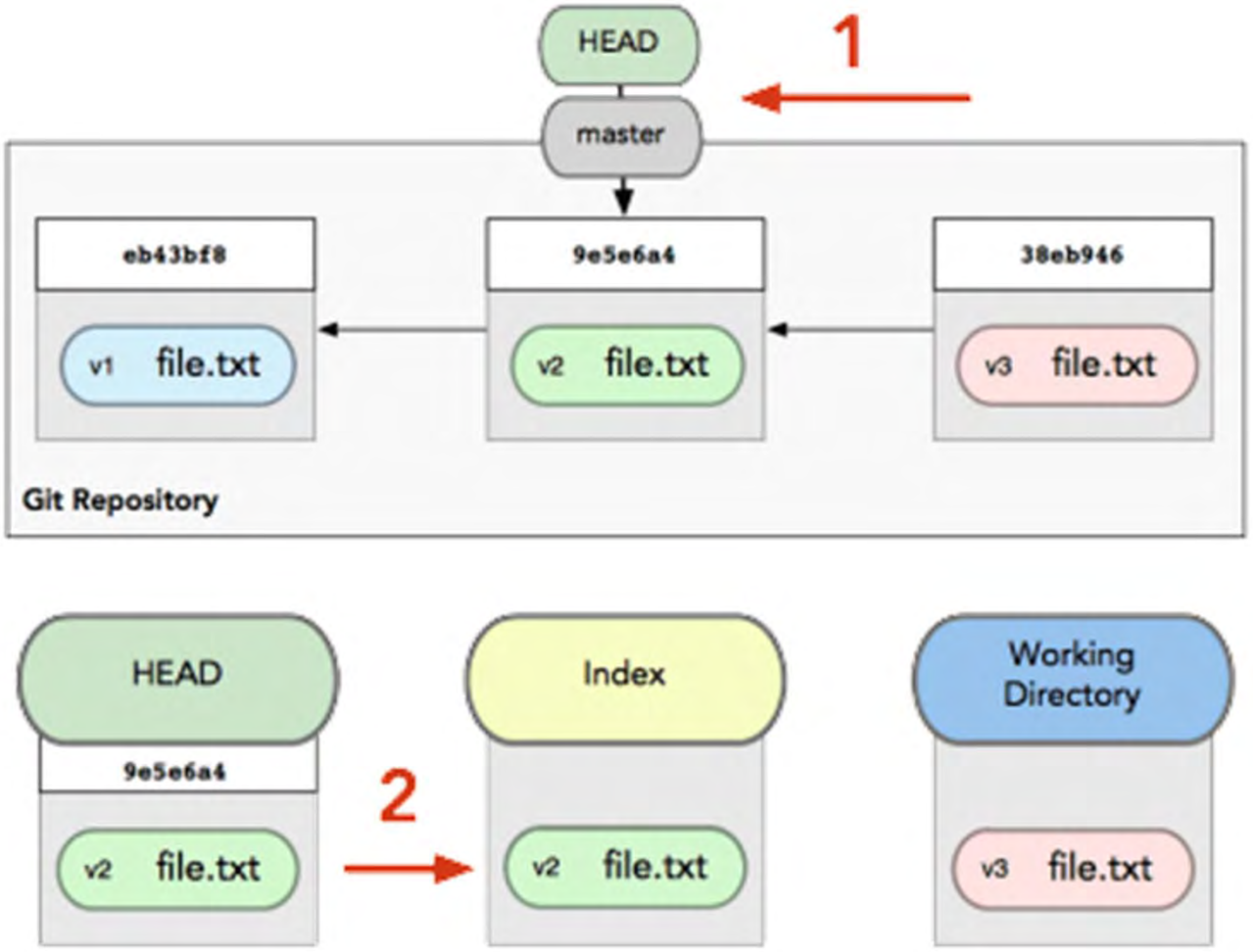
GIT – Área de Trabajo

GIT – Área de Trabajo



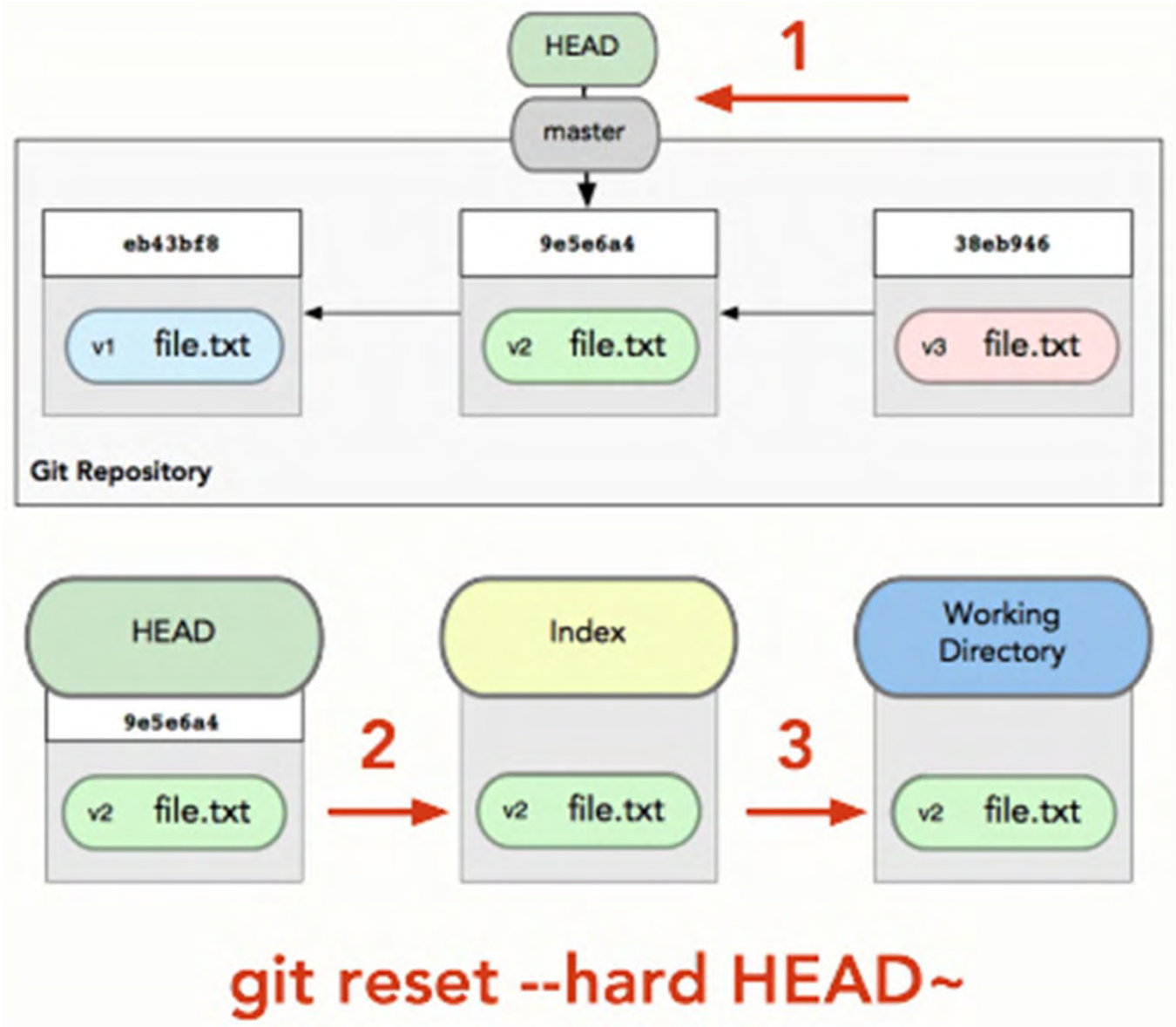
git reset --soft HEAD~

GIT – Área de Trabajo

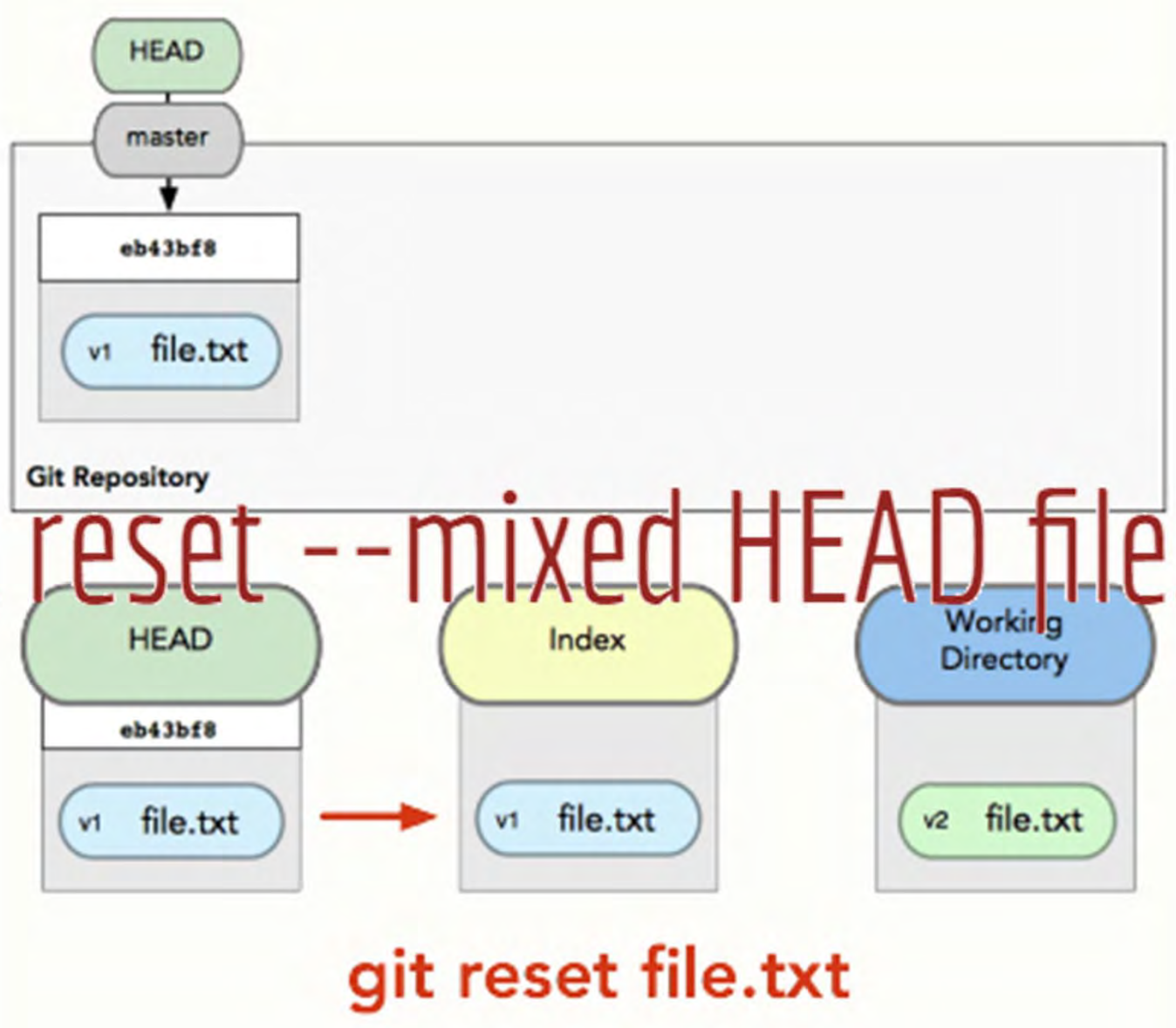


git reset [--mixed] HEAD~

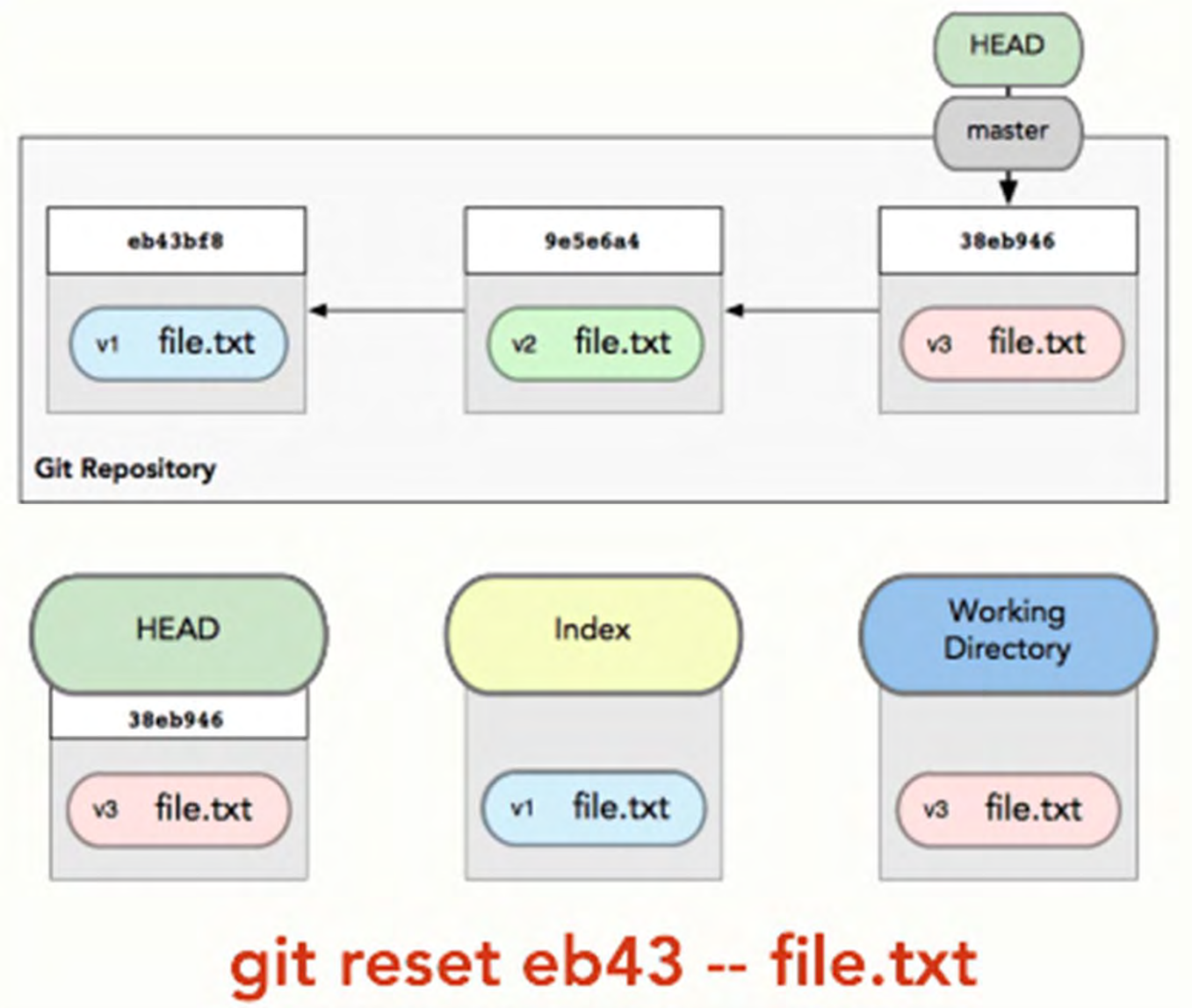
GIT – Área de Trabajo



GIT – Área de Trabajo



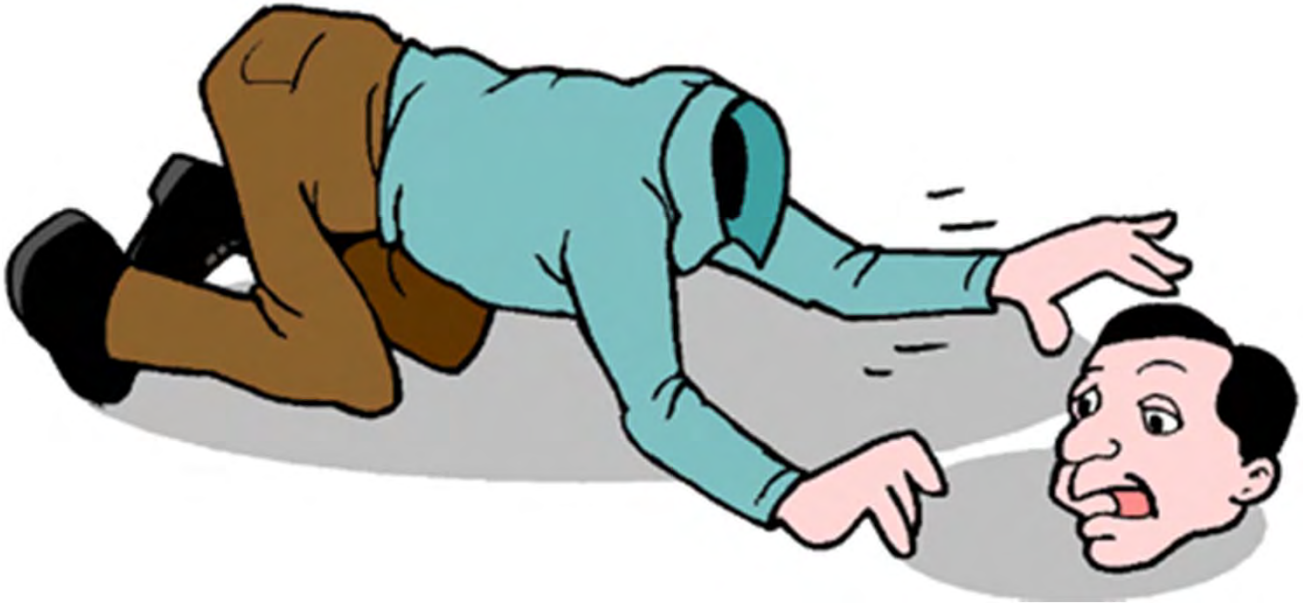
GIT – Área de Trabajo



GIT – Área de Trabajo

NETT FORMACIÓN

Don't Lose Your Head



GIT – Área de Trabajo

NETT FORMACIÓN

But ...

GIT – Área de Trabajo

NETT FORMACIÓN

But ...

git reset HEAD@{n}

GIT – Área de Trabajo

NETT FORMACIÓN

But ...

git hist --all

GIT – Área de Trabajo

What is a rebase

Rebasing is the process of moving a branch to a new base commit.

Git moves a branch from one commit to another.

Git creates new commits applying them to the base.

git What is a rebase

NETT FORMACIÓN

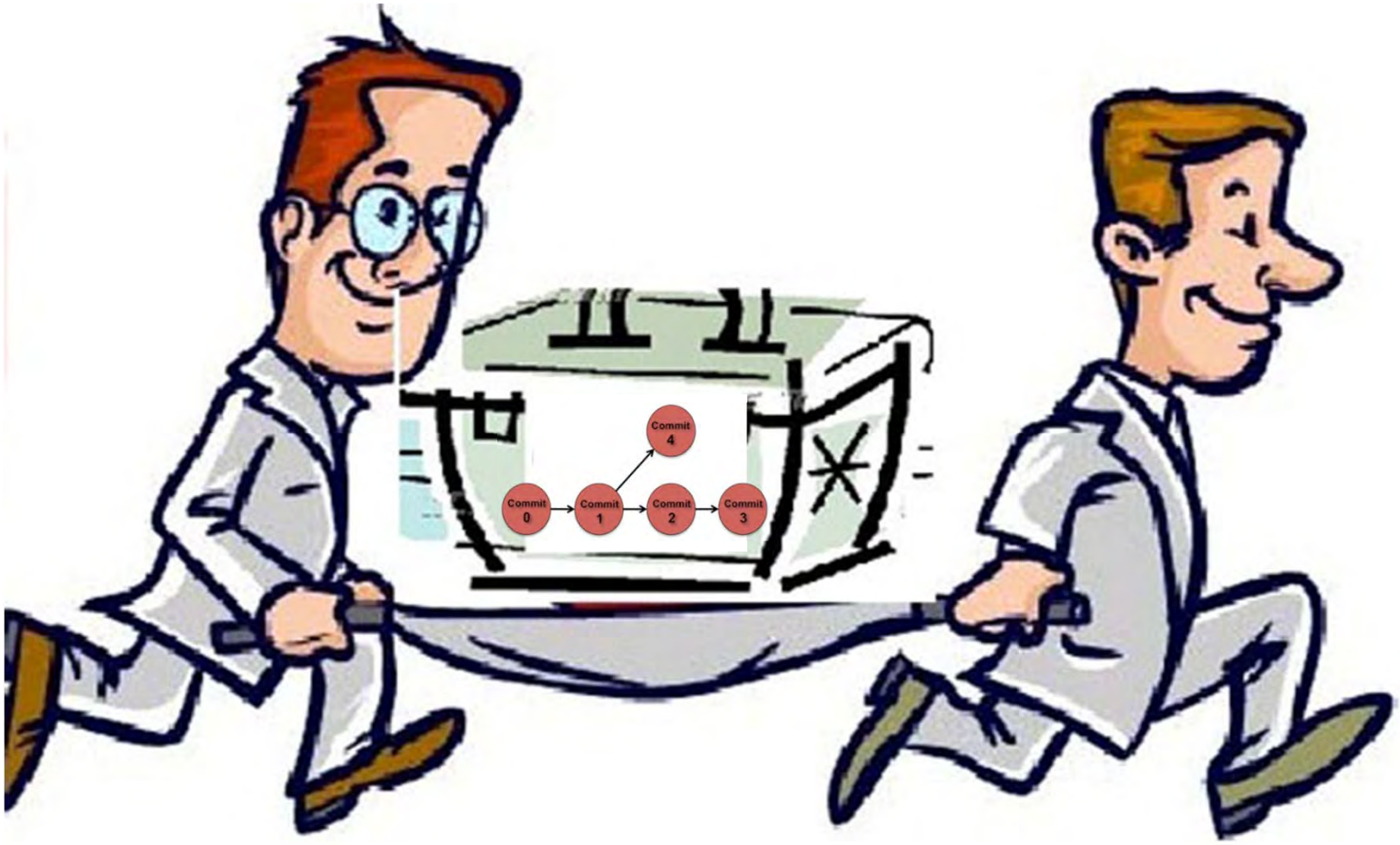
The git rebase command is used to alter where a sequence of commits is based.

A rebase helps to cut up commits and slice them into any way and placed exactly where you want them.

This command requires at least the name of the other branch onto which your commits will be relocated.

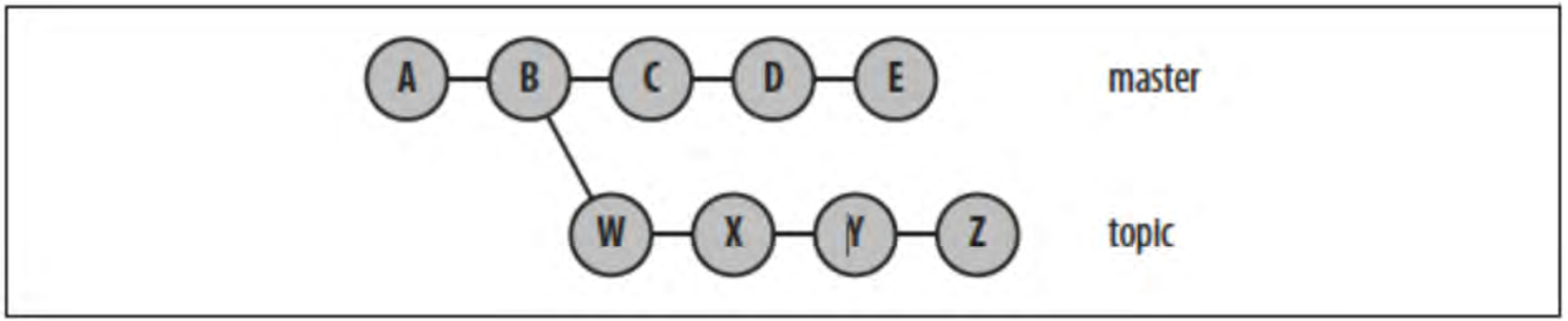
By default, the commits from the current branch that are not already on the other branch are rebased.

GIT – Área de Trabajo



NETT FORMACIÓN

Before git rebase



Rebasing commits

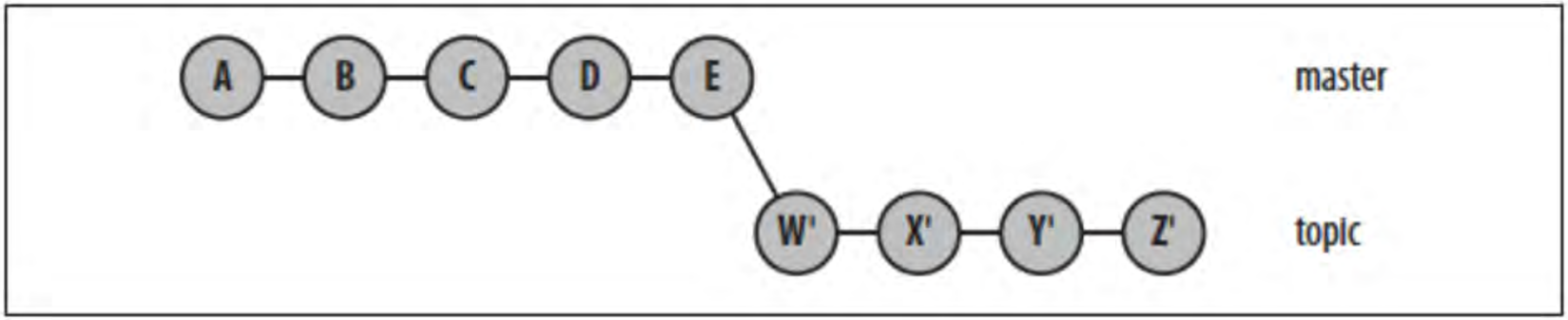

```
$ git checkout topic
$ git rebase master
```

or:

```
$ git rebase master topic
```

NETT FORMACIÓN

After git rebase



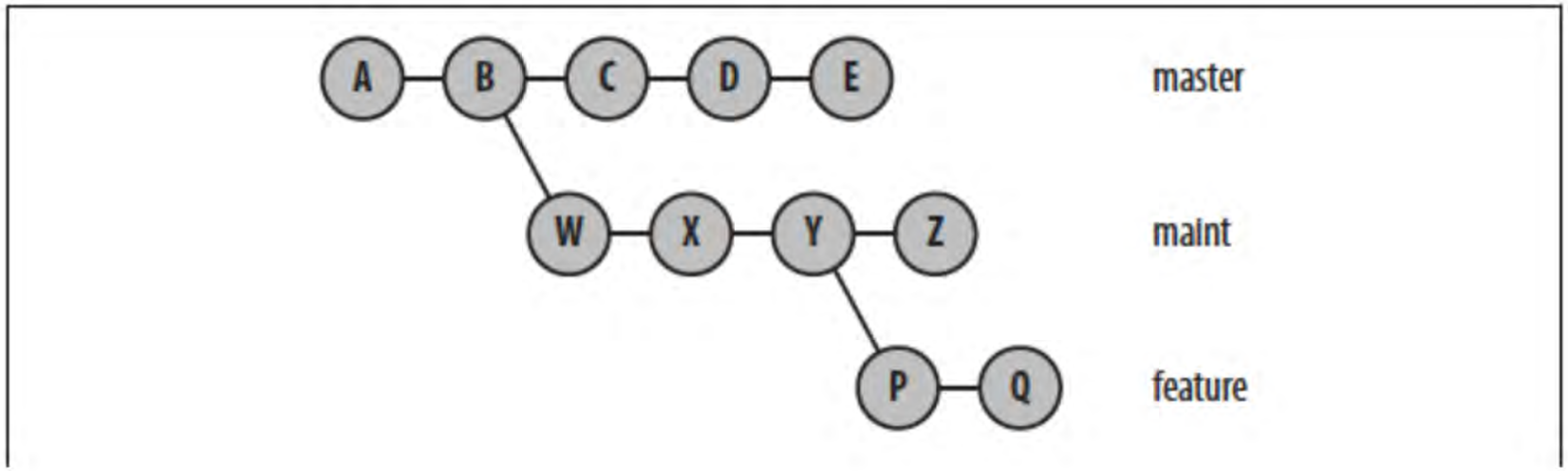

```
$ git checkout topic
$ git rebase master
```

or:

```
$ git rebase master topic
```

NETT FORMACIÓN

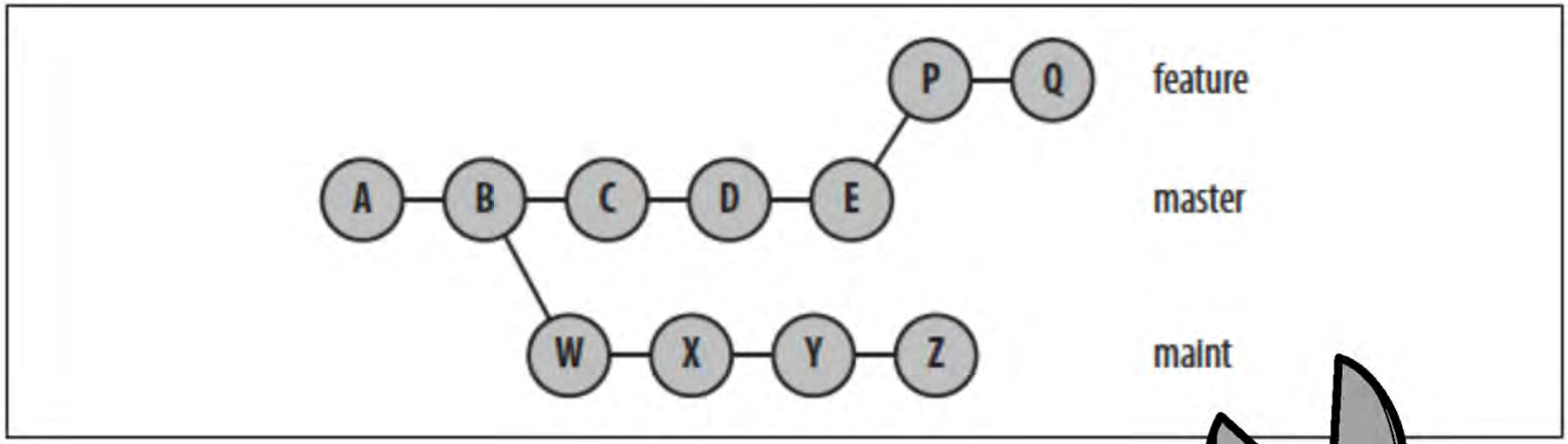
Before git rebase transplant



git rebase --onto master maint^ feature

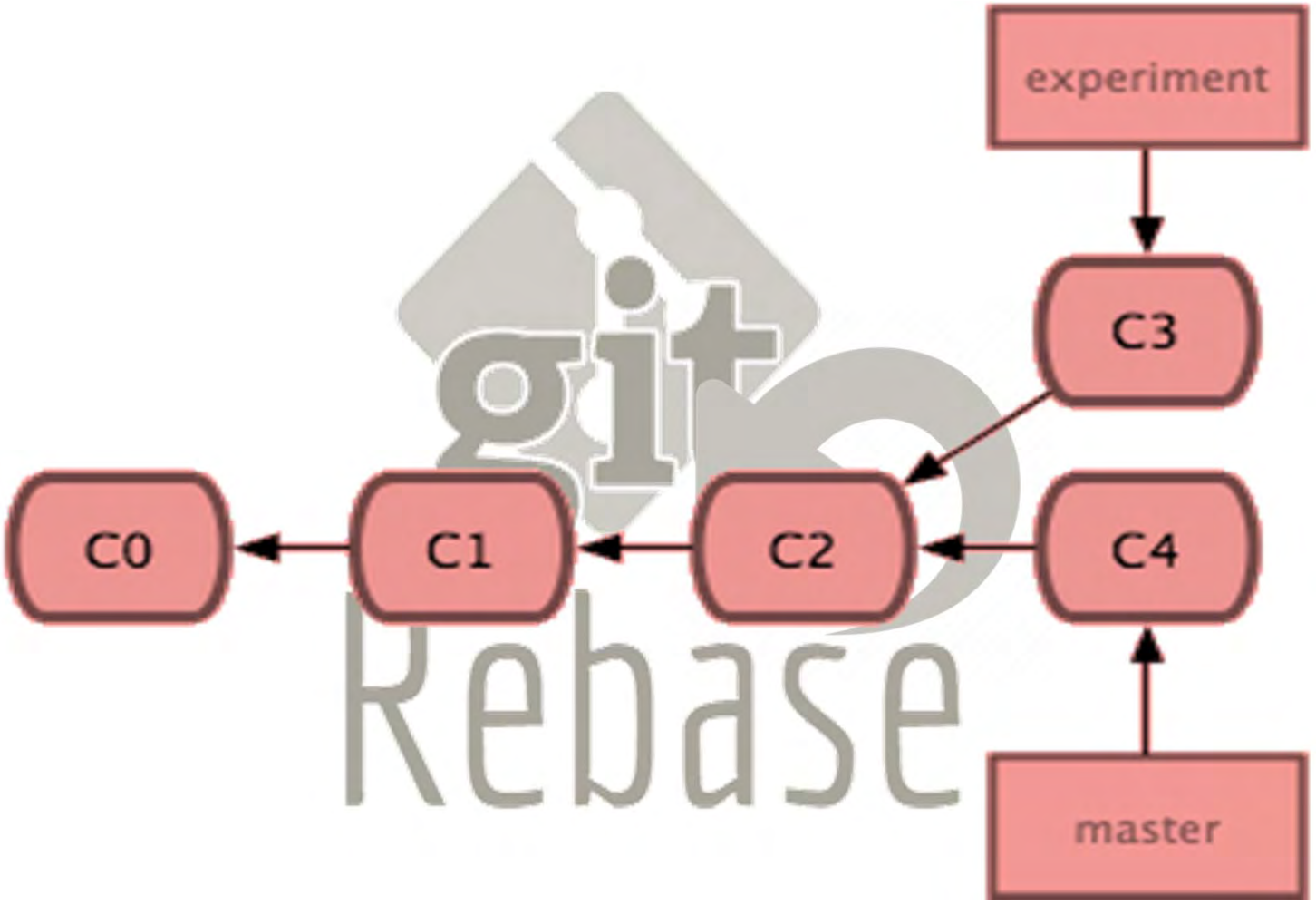
NETT FORMACIÓN

After git rebase transplant

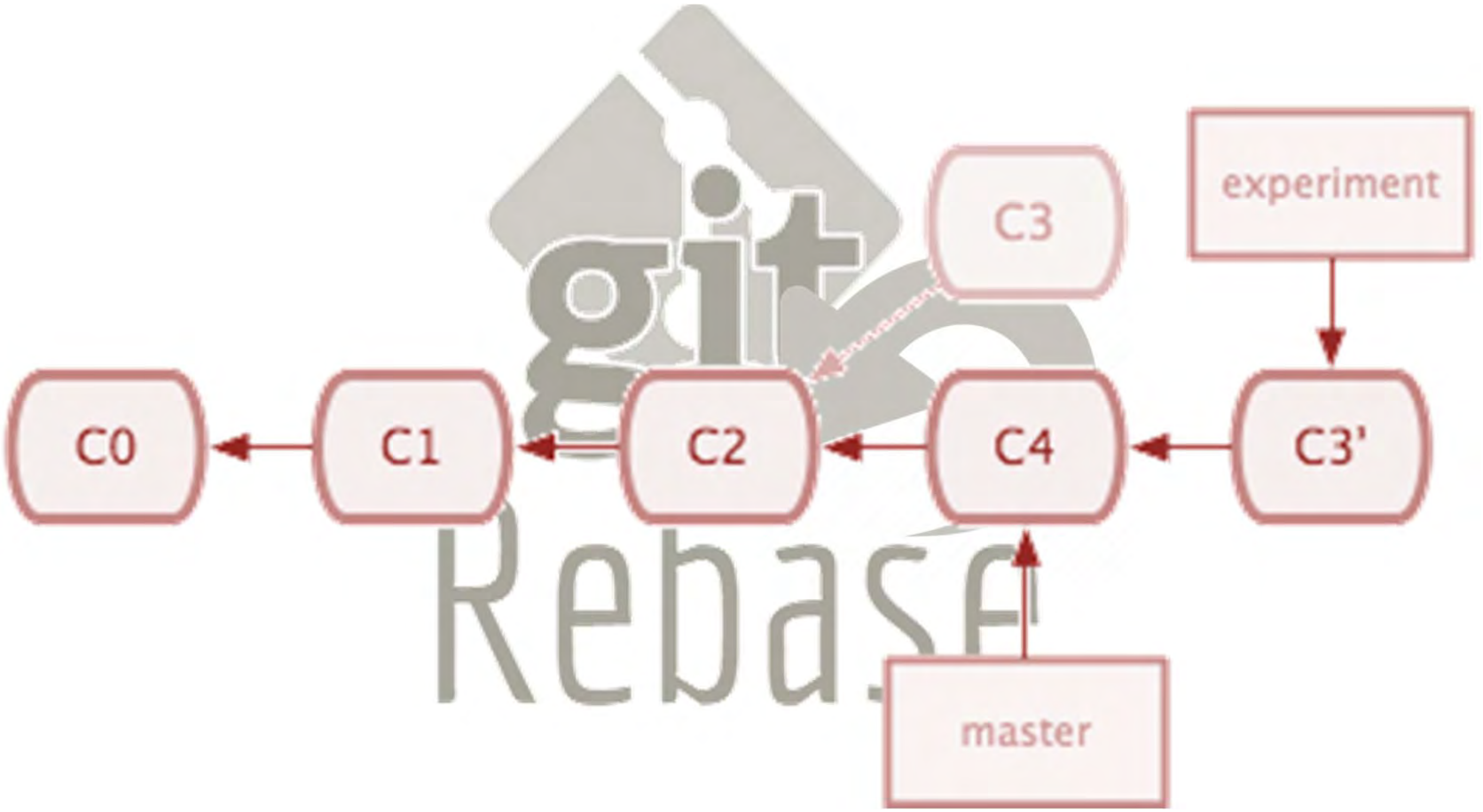


git rebase --onto master maint^ feature

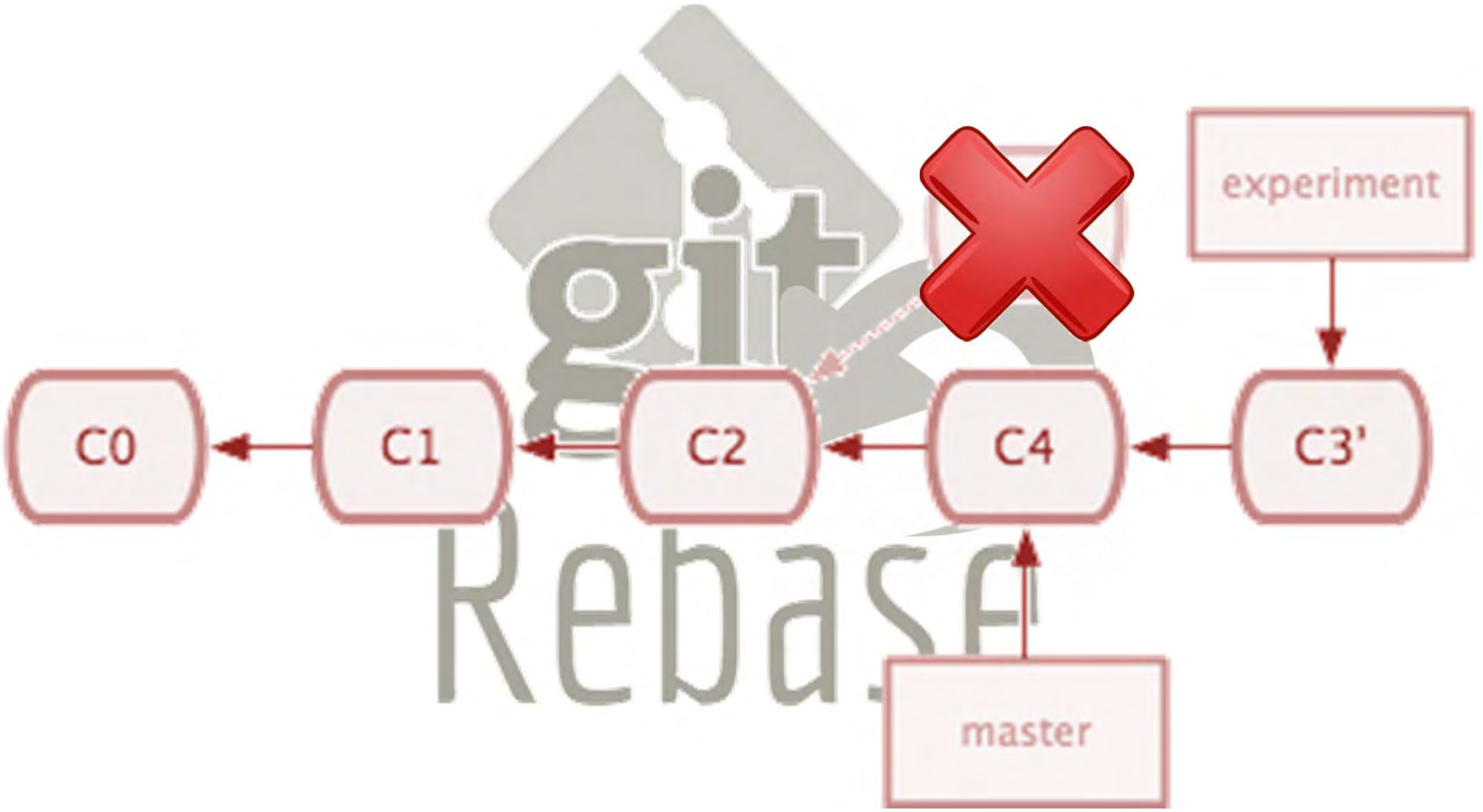
GIT – Área de Trabajo



GIT – Área de Trabajo



GIT – Área de Trabajo



GIT – Área de Trabajo

If a conflict is found, the rebase operation suspends its processing temporarily so you can resolve the conflict and :

rebase --continue: continue to the next commit.

rebase --skip: skip this commit and move to the next.

rebase –abort: abandons the operation and restores the repository to the state prior to issuing the original git rebase.

GIT – Área de Trabajo

NETT FORMACIÓN

Merging VS rebasing

Result of the rebase command looks much like that of the merge. The style branch currently contains all its changes, plus all the changes of the master branch. The commit tree, however, is a bit different. The style branch commit tree has been rewritten to make the master branch a part of the commit history. This makes the chain of commits linear and more readable.

When to use the rebase, when the merge?

Don't use the rebase command …

1. If the branch is public and shared. Rewriting such branches will hinder the work of other team members.
2. When the *exact* commit branch history is important (because the rebase command rewrites the history of commits).

Given the above recommendations, I prefer to use rebase for short-term, local branches and the merge command for branches in the public repository.

GIT – Área de Trabajo

NETT FORMACIÓN

Merging VS rebasing

Result of the rebase command looks much like that of the merge. The style branch currently contains all its changes, plus all the changes of the master branch. The commit tree, however, is a bit different. The style branch commit tree has been rewritten to make the master branch a part of the commit history. This makes the chain of commits linear and more readable.

When to use the rebase, when the merge?

Don't use the rebase command …

1. If the branch is public and shared. Rewriting such branches will hinder the work of other team members.
2. When the *exact* commit branch history is important (because the rebase command rewrites the history of commits).

Given the above recommendations, I prefer to use rebase for short-term, local branches and the merge command for branches in the public repository.

GIT – Área de Trabajo

NETT FORMACIÓN

Merging VS rebasing

Result of the rebase command looks much like that of the merge. The style branch currently contains all its changes, plus all the changes of the master branch. The commit tree, however, is a bit different. The style branch commit tree has been rewritten to make the master branch a part of the commit history. This makes the chain of commits linear and more readable.

When to use the rebase, when the merge?

Don't use the rebase command …

1. If the branch is public and shared. Rewriting such branches will hinder the work of other team members.
2. When the *exact* commit branch history is important (because the rebase command rewrites the history of commits).

Given the above recommendations, I prefer to use rebase for short-term, local branches and the merge command for branches in the public repository.

GIT – Área de Trabajo

git Cherry picks

NETT FORMACIÓN

The command git cherry-pick commit applies the changes introduced by the named commit on the current branch.

It will introduce a **new, distinct commit**.

Strictly speaking, using git cherry-pick doesn't alter the existing history within a repository; instead, it adds to the history.

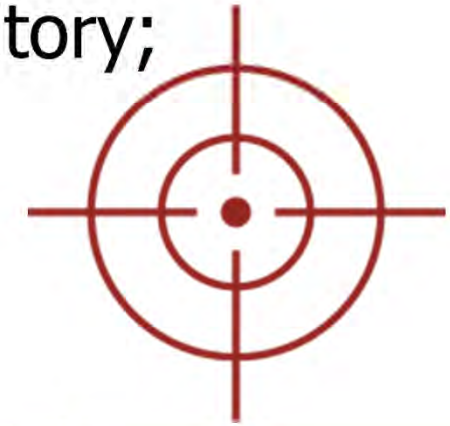
git Cherry picks

NETT FORMACIÓN

The command git cherry-pick commit applies the changes introduced by the named commit on the current branch.

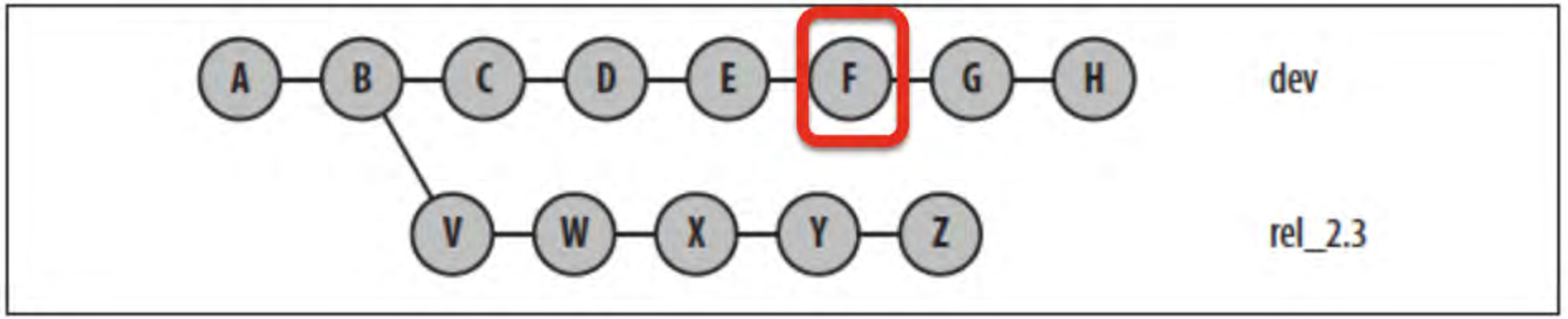
It will introduce a **new, distinct commit**.

Strictly speaking, using git cherry-pick doesn't alter the existing history within a repository; instead, it adds to the history.



NETT FORMACIÓN

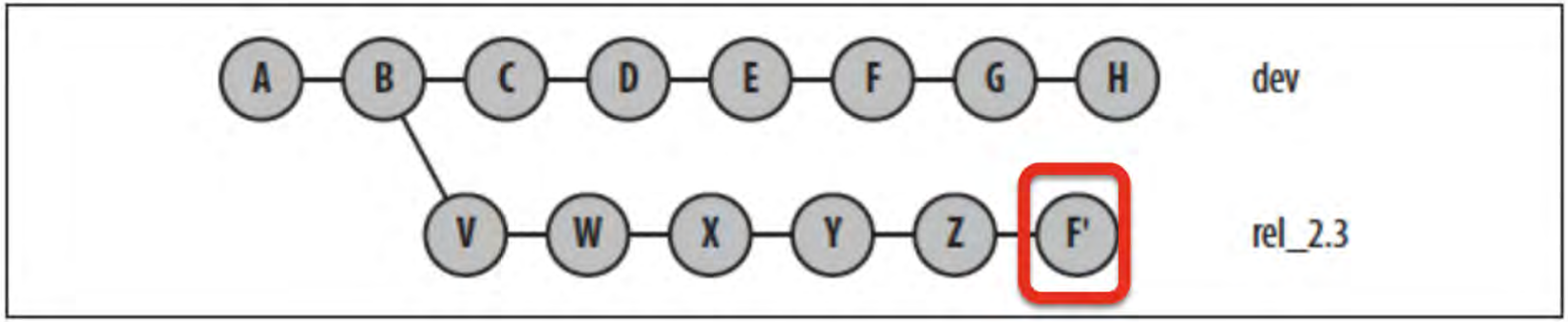
Before



We want to add F to rel_2.3 branch

NETT FORMACIÓN

After



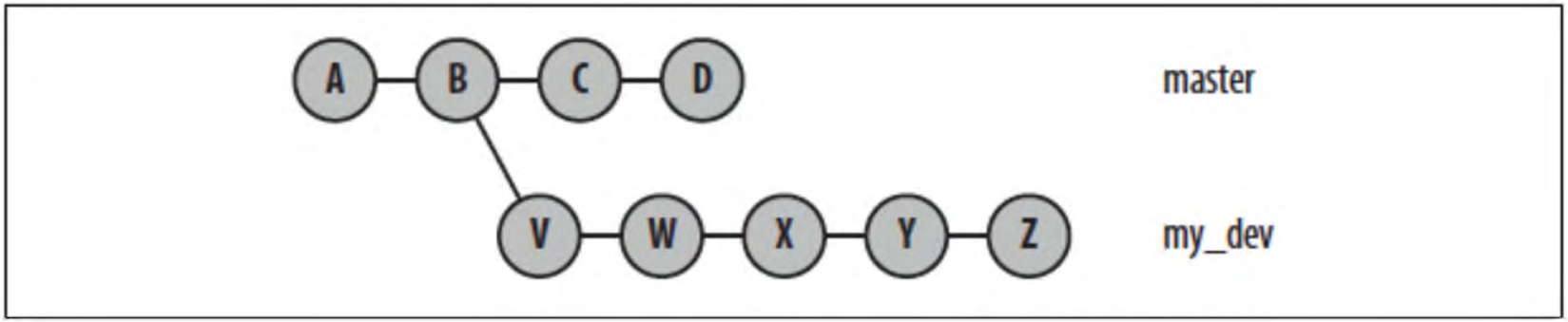

```
$ git checkout rel_2.3
$ git cherry-pick dev~2
```

git Cherry picks

```
$ git checkout master
$ git cherry-pick my_dev^     # Y
$ git cherry-pick my_dev~3    # W
$ git cherry-pick my_dev~2    # X
$ git cherry-pick my_dev      # Z
```

Alter the order

You can alter the order of commits when you apply cherry picks.



New order to master branch Y, W, X, Z


Git Remote
Tags

A remote repository is a reference to another repository.

A remote is a shorthand name for a Git URL.

You can define any number of remotes in a repository

Once a remote is established, Git can transfer data from one repository to another using push and pull model.

To keep track of data from other repositories, Git uses tracking branches.

Each tracking branch in your repository is a local branch that serves as a proxy for a specific branch in a remote repository.

Remotes

$ git remote

add: adds a remote

rm: deletes a remote

rename: rename a remote repo

-v: list all remotes

GIT – Remotos

Adding a remote repository

```
$ git remote add origin https://github.com/user/repo.git
# Set a new remote

$ git remote -v
# Verify new remote
origin  https://github.com/user/repo.git (fetch)
origin  https://github.com/user/repo.git (push)
```

GIT – Remotos

Add a local branch tracking the remote branch

```
$ git branch --track style origin/style
Branch style set up to track remote branch style from origin.
$ git branch -a
  style
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/style
  remotes/origin/master
$ git hist --max-count=2
* 2faa4ea 2011-03-09 | Changed README in original repo (HEAD, origin/master, origin/HEAD, master)
[Alexander Shvets]
* 6e6c76a 2011-03-09 | Updated index.html (origin/style, style) [Alexander Shvets]
```

GIT – Remotos

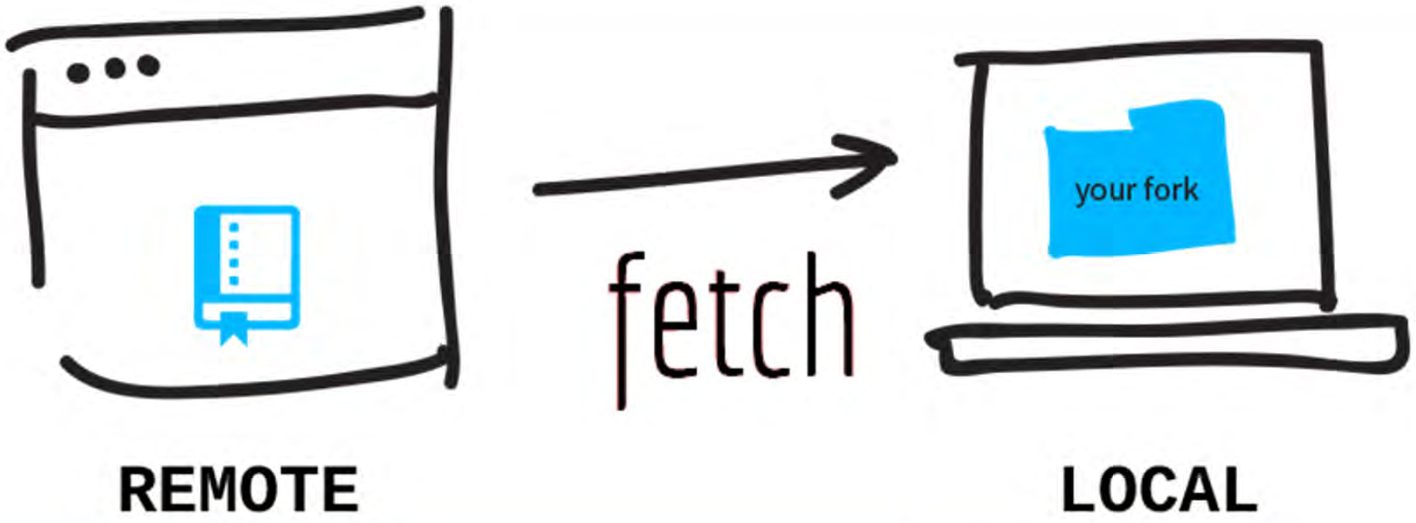
Add a local branch tracking the remote branch

```
$ git branch --track style origin/style
Br[illegible] style [illegible]t up to track remote branch style from origin.
$ git branch -a
  style
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/style
  remotes/origin/master
$ git hist --max-count=2
* 2faa4ea 2011-03-09 | Changed README in original repo (HEAD, origin/master, origin/HEAD, master)
[Alexander Shvets]
* 6e6c76a 2011-03-09 | Updated index.html (origin/style, style) [Alexander Shvets]
```

In addition to git clone , other common Git commands that refer to remote repositories are:

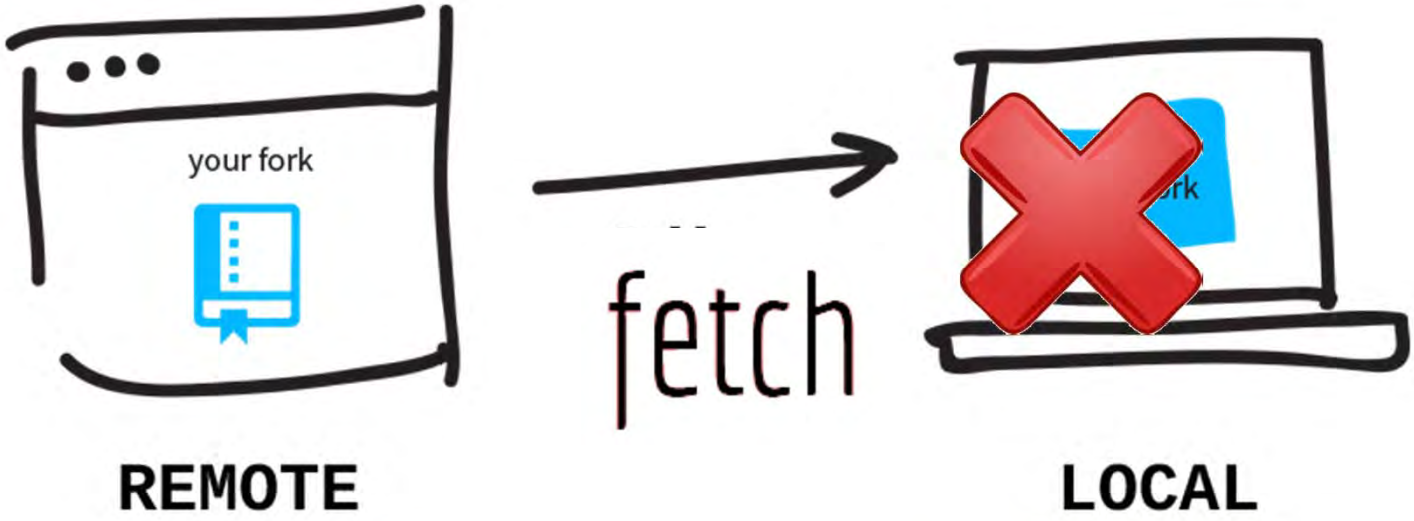
- git fetch: Retrieves objects and their related metadata from a remote repository

```
$ git fetch remotename
# Fetches updates made to a remote repository
```



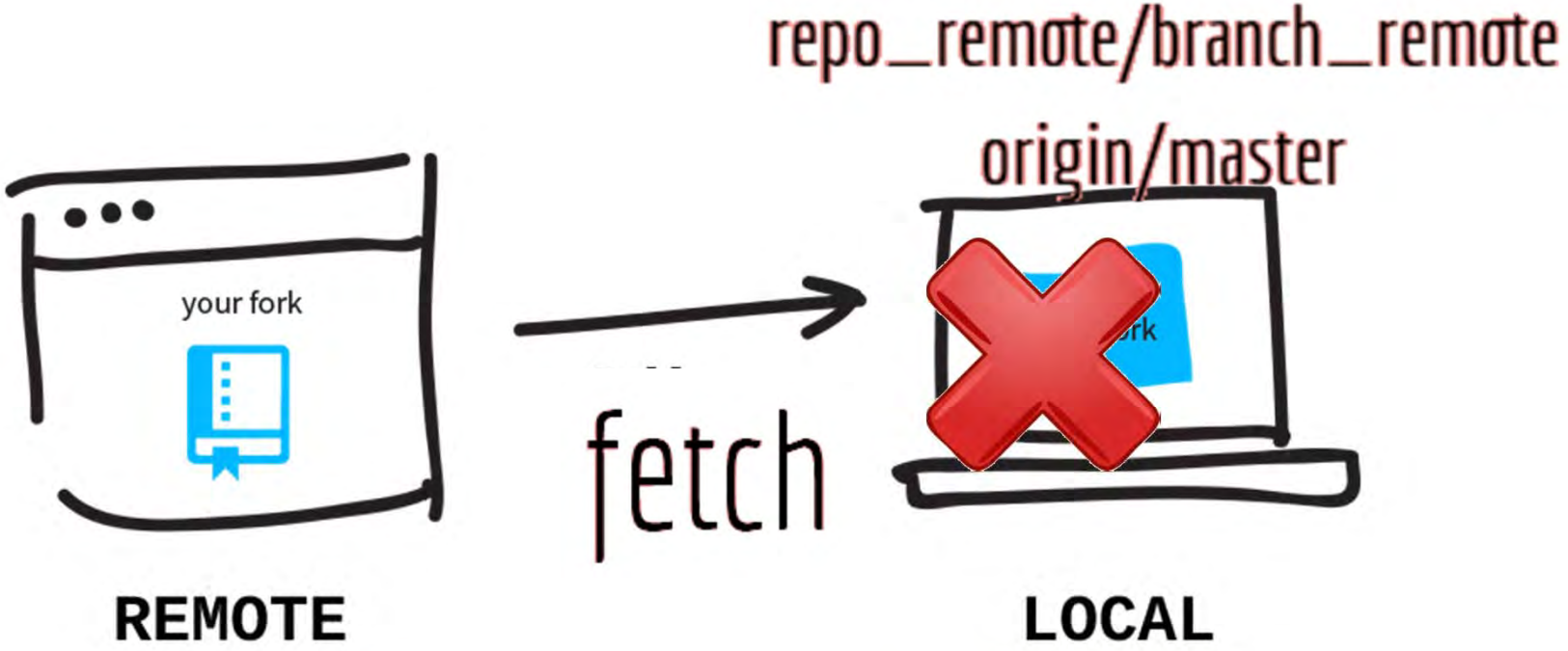
In addition to git clone , other common Git commands that refer to remote repositories are:

- git fetch: Retrieves objects and their related metadata from a remote repository



In addition to git clone , other common Git commands that refer to remote repositories are:

- git fetch: Retrieves objects and their related metadata from a remote repository




Git Remote
Tags
Git pull

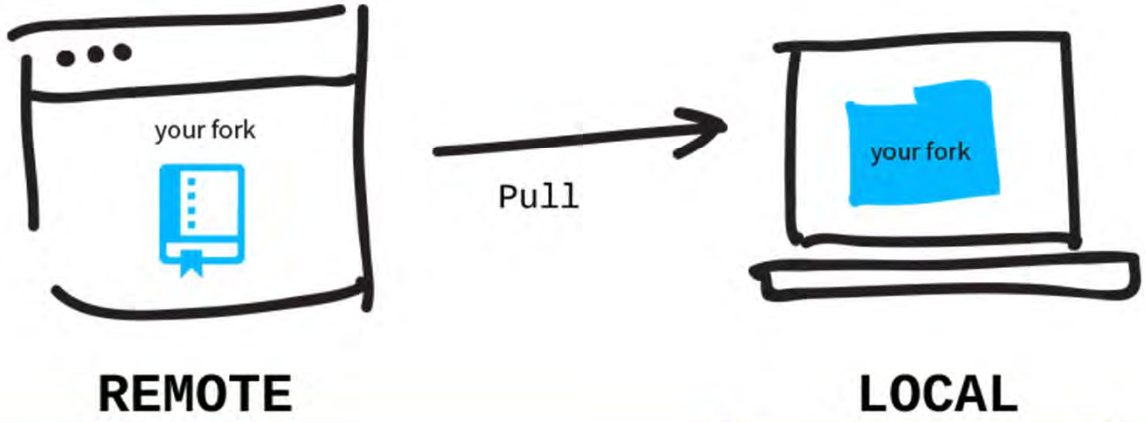
git Remotes

In addition to git clone , other common Git commands that refer to remote repositories are:

- git fetch: Retrieves objects and their related metadata from a remote repository

- git pull: Fetch + Merge

```
$ git pull remotename branchname
# Grabs online updates and merges them with your local work
```



git Remotes

In addition to git clone , other common Git commands that refer to remote repositories are:

- git fetch: Retrieves objects and their related metadata from a remote repository
- git pull: Fetch + Merge
- git push: Transfers objects and their related metadata to a remote repository


git push

NETT FORMACIÓN

Push

$ git push <repo> <branch>

git push remote_repo [remote_branch]

git push origin develop

git Push to a remote branches

NETT FORMACIÓN

Push

$ git push <repo> <branch>

Force Push

$ git push –f <repo> <branch>

¡Be careful forcing the push!

GIT – Remotos

NETT FORMACIÓN

$ git push <repo> --tags: Sends all tags

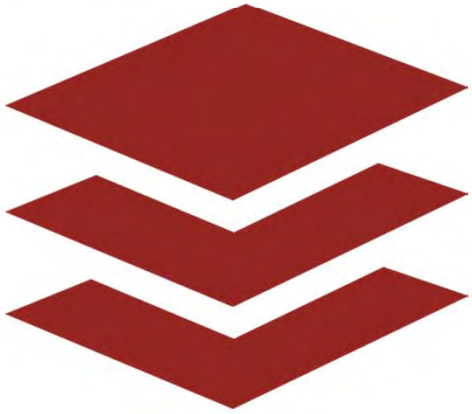
$ git push <repo> <tag-name>: Sends the tag

git Remotes

In addition to git clone , other common Git commands that refer to remote repositories are:

- git fetch: Retrieves objects and their related metadata from a remote repository
- git pull: Fetch + Merge
- git push: Transfers objects and their related metadata to a remote repository
- git ls-remote: Shows references within a remote

Stash



An intermediate area where you can commit unstable code that you can not deal with.

Use stash when you want to record the current state of the working directory and the index, but want to go back to a clean working directory.

git stashing

A great way to pause what you are currently working on and come back to it later.

$ git stash: Stash your changes away

$ git stash apply: Bring work back

E.g: git stash apply stash@{1}

$ git stash list: List of stashes

$ git stash pop: apply the top stash

$ git stash drop <id>: Manually delete stashes

$ git stash clear: Delete al of the stored stashes.

git Commands

- git stash list
- git stash save "mensaje”
- git stash pop
- git stash show stash@{0}
- git stash apply stash@{0}
- git diff stash@{0}


BLaMe

git

NETT FORMACIÓN

Git Blame

Tells you who last modified each line of a file and which commit made the change:

$ git blame <file>

```
mamoghli@MAMOGHLI2 MINGW64 /c/HR-Personal/AAA/Repo_AAA (master)
$ git blame doc1.txt
e9b639b3 (Noel Mamoghli 2017-01-26 15:58:58 +0100 1) I'm the Doc One!
a923c3c7 (Noel Mamoghli 2017-01-26 16:22:57 +0100 2) Name: Thrawn

mamoghli@MAMOGHLI2 MINGW64 /c/HR-Personal/AAA/Repo_AAA (master)
$
```

NETT FORMACIÓN

DIFFS

NETT FORMACIÓN

A compact summary of the differences between two items.

Diffs can help you make well-structured commits during your normal development process.

NETT FORMACIÓN

There are three basic sources to use git diff:

- Any object (commit, branch name, tag, etc) anywhere within the entire commit graph.
- Your working directory.
- The index

NETT FORMACIÓN

$ git diff

Shows the difference between your working directory and the index.

Exposes what is dirty in your working directory.

Does not reveal differences between what's in your index and what is permanently stored in the repository.

NETT FORMACIÓN

$ git diff

Shows the difference between your working directory and the index.

Exposes what is dirty in your working directory.

Does not reveal differences between what's in your index and what is permanently stored in the repository.

NETT FORMACIÓN

$ git diff <commit>

Summarizes differences between your working directory and the given commit.

<commit> can be HEAD or a particular branch name.

NETT FORMACIÓN

$ git diff --cached <commit>

Shows the differences between index and the given commit.

Default <commit> value is HEAD that shows you how your next commit will alter the current branch.

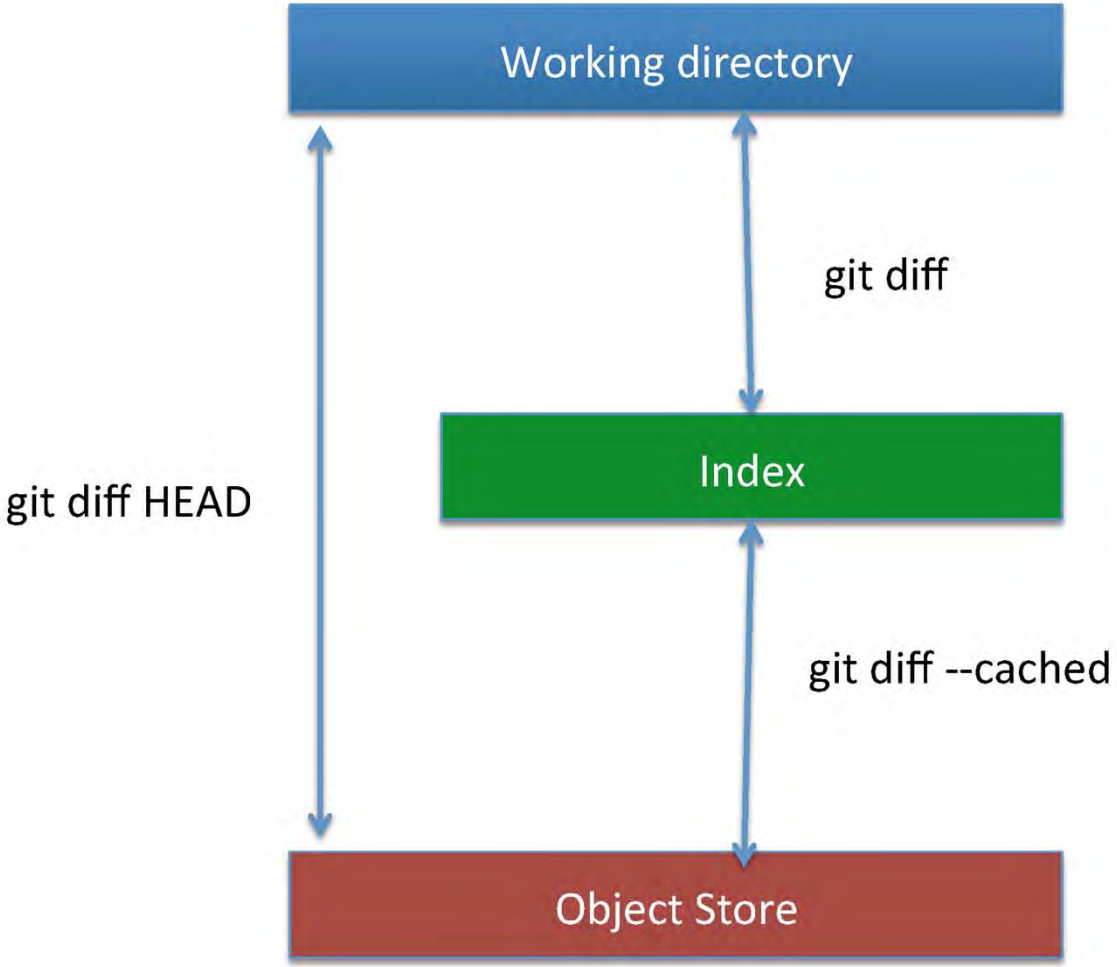
$ git diff <commit1> <commit2>

$ git diff <commit1>..<commit2>

Differences between the two commits.

This command ignores the index and working directory.

Forms of the git diff



NETT FORMACIÓN

$ git diff <commit1> <commit2> <file>
$ git diff <commit1>..<commit2> <file>

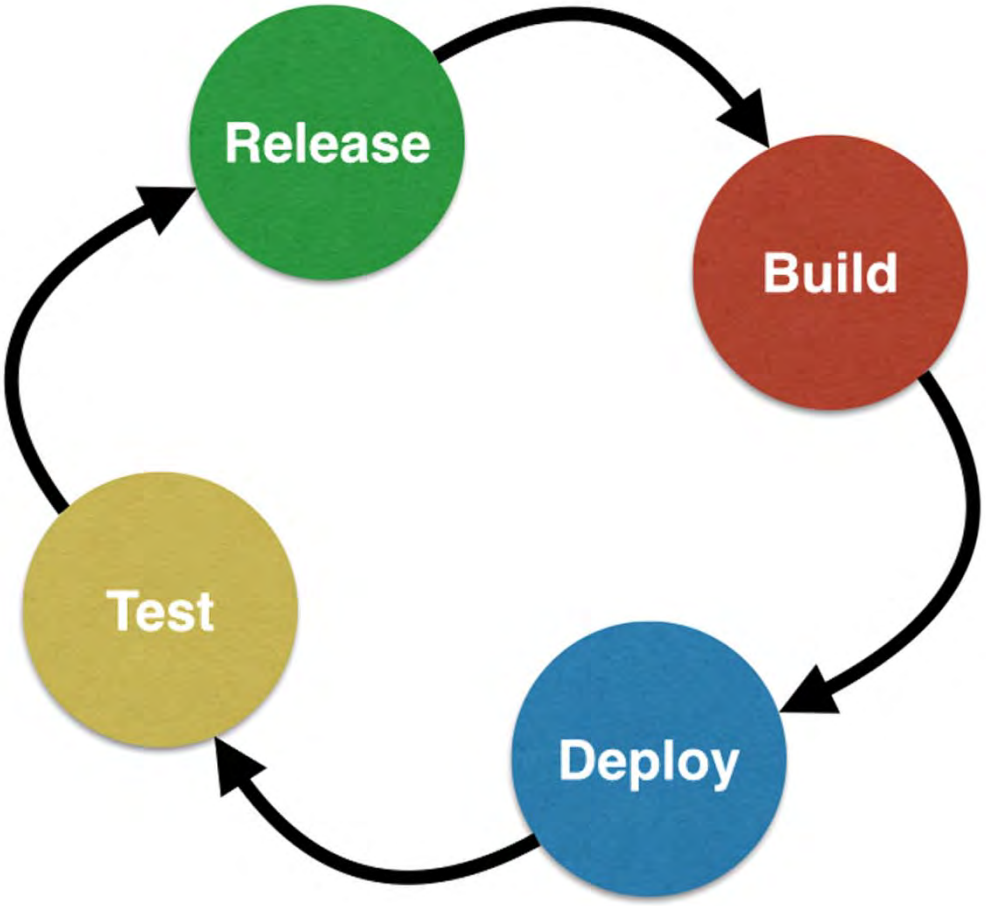
The file differences between the two commits.

This command ignores the index and working directory.

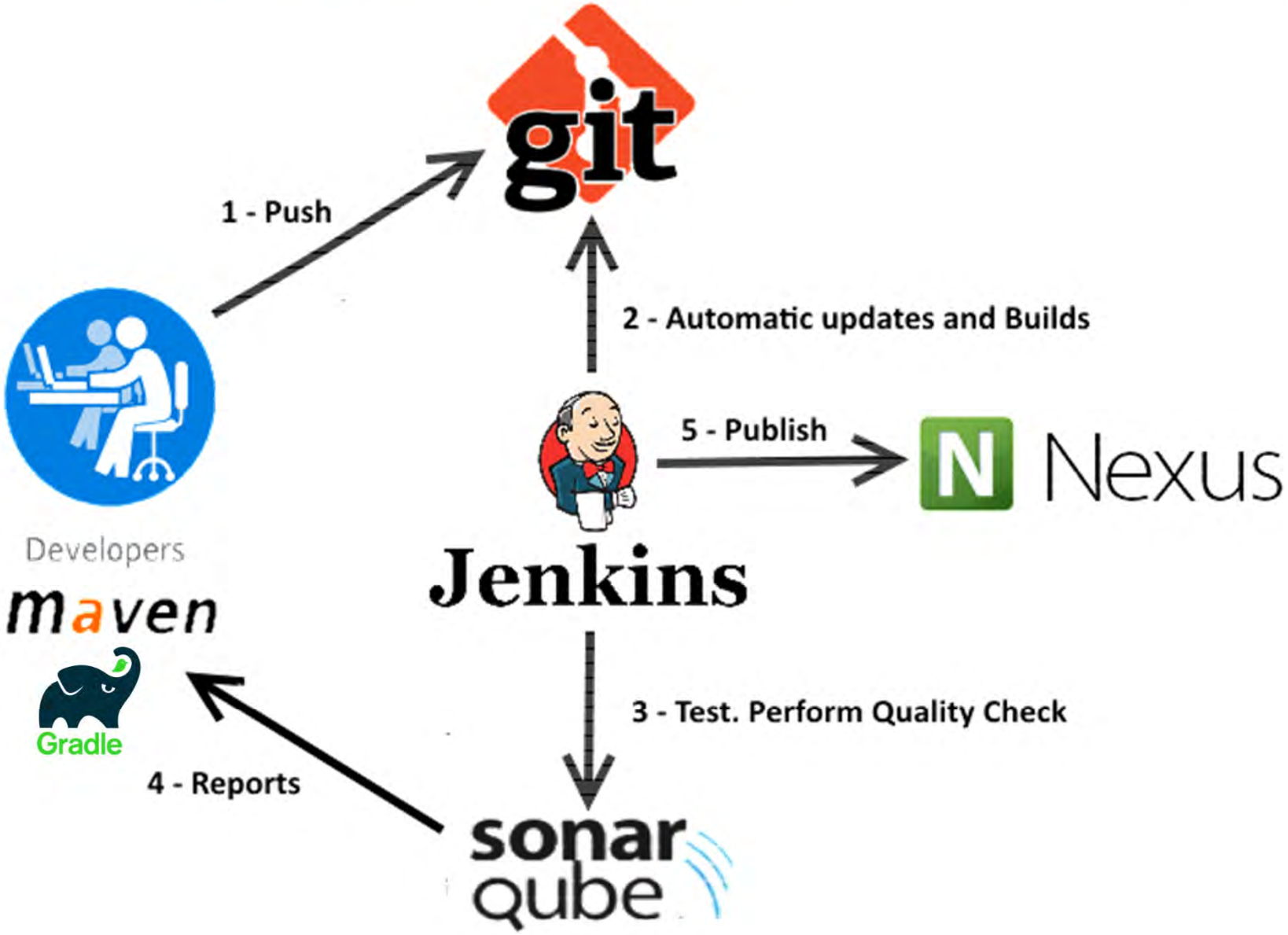
git

NETT FORMACIÓN

Integración Continua

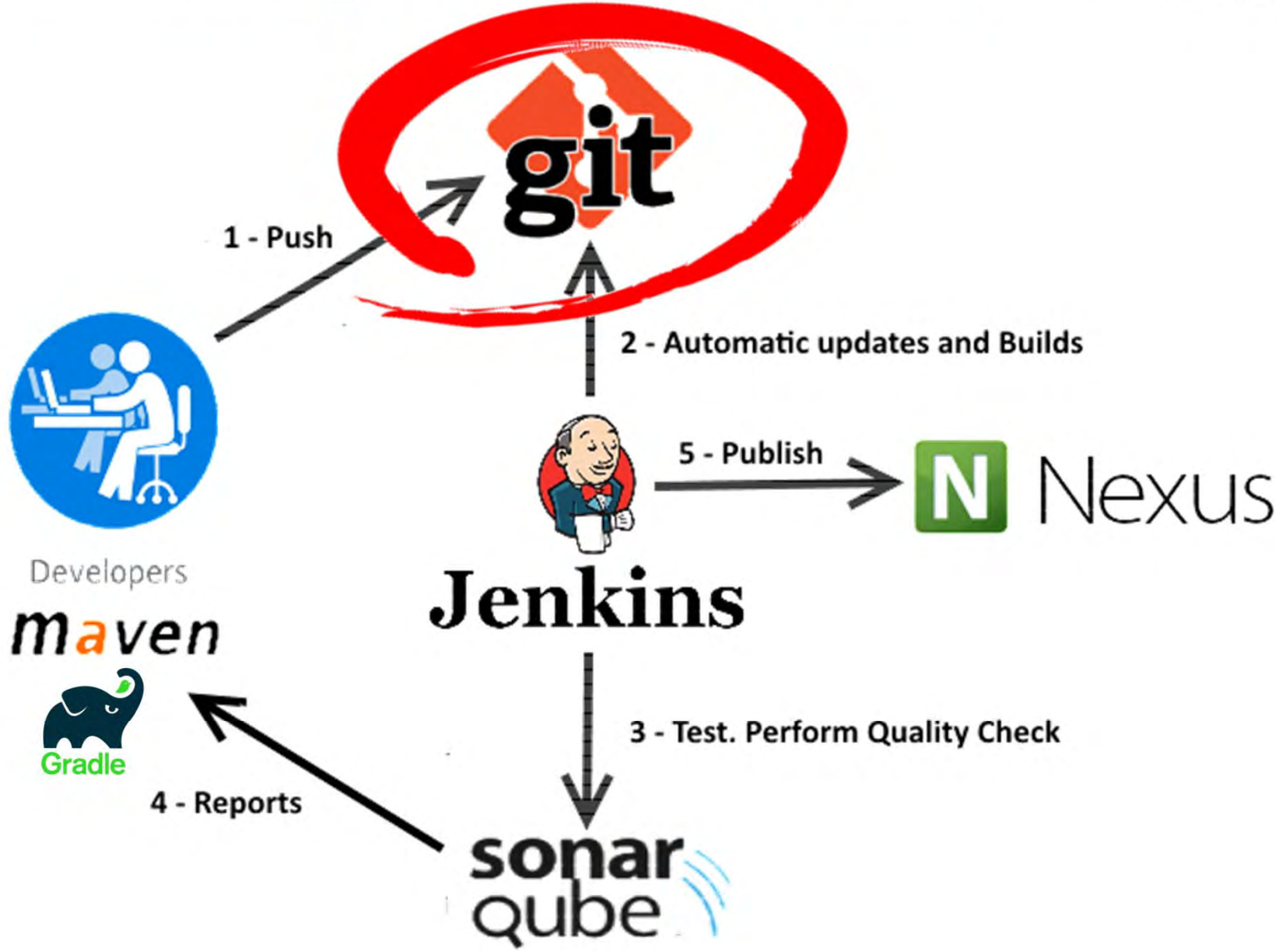


GIT – Integración Continua



GIT – Integración Continua

NETT FORMACIÓN



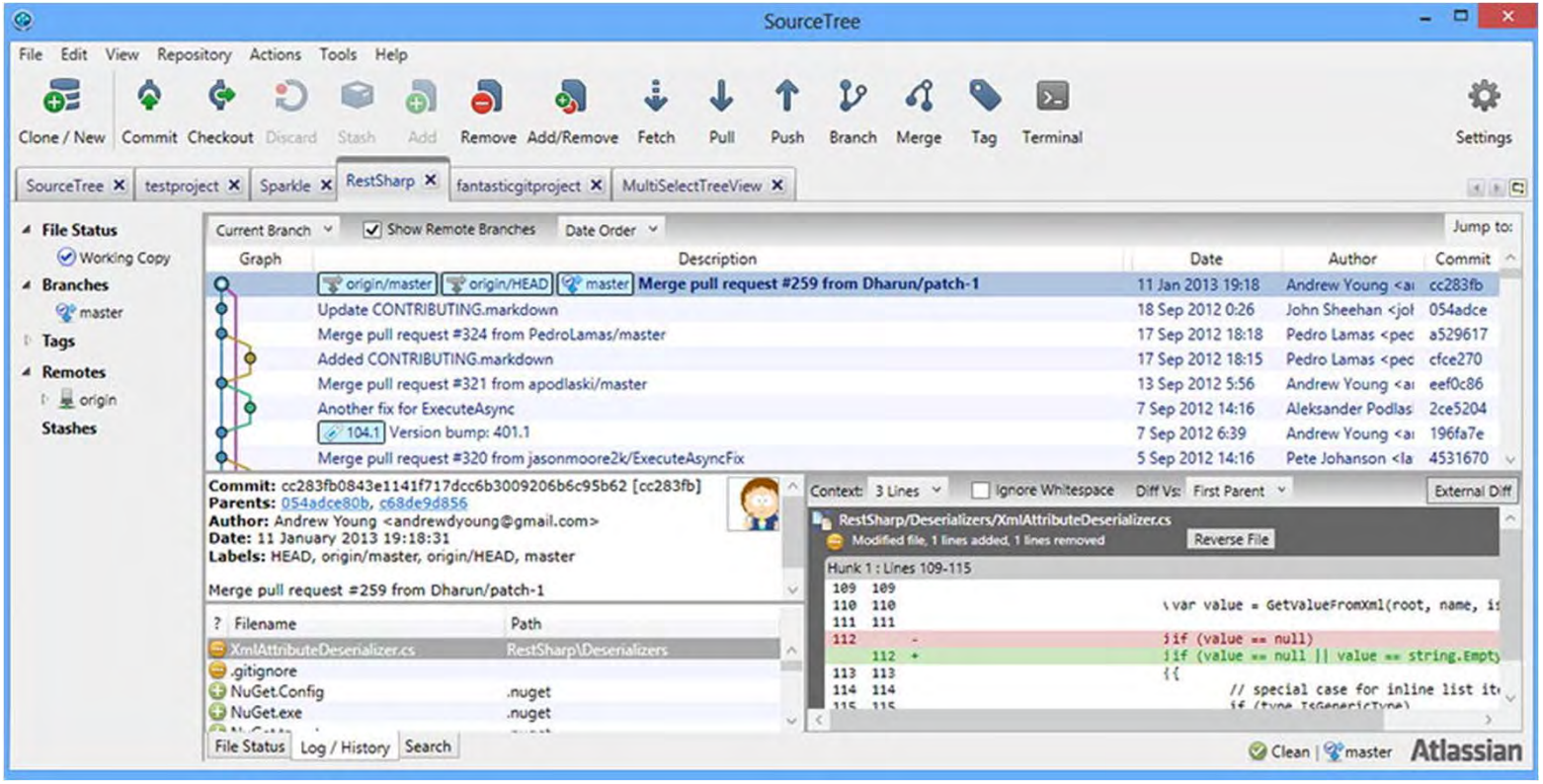
git

NETT FORMACIÓN

Clientes Gráficos

GIT – Clientes Gráficos

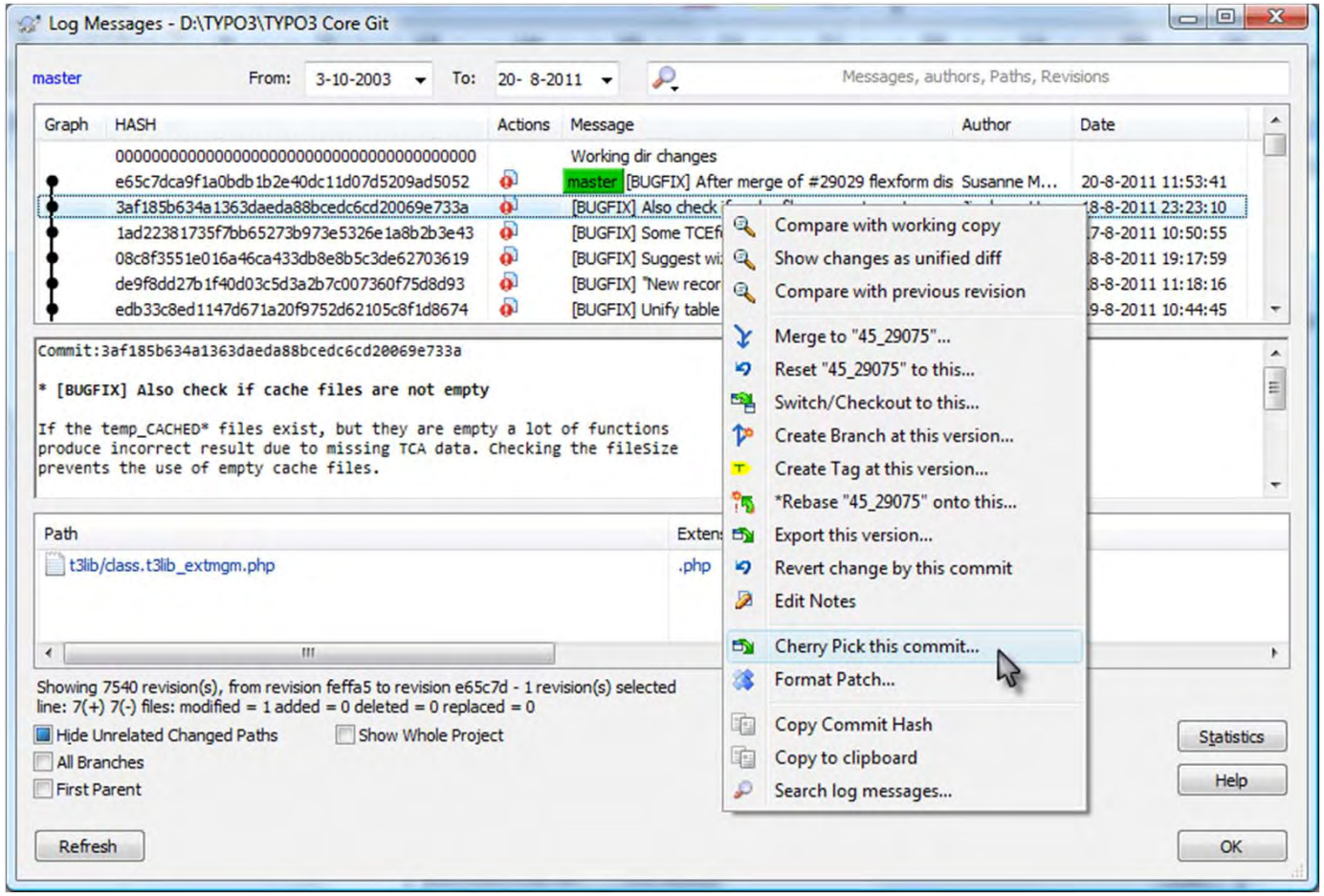
SourceTree



https://www.sourcetreeapp.com/

GIT – Clientes Gráficos

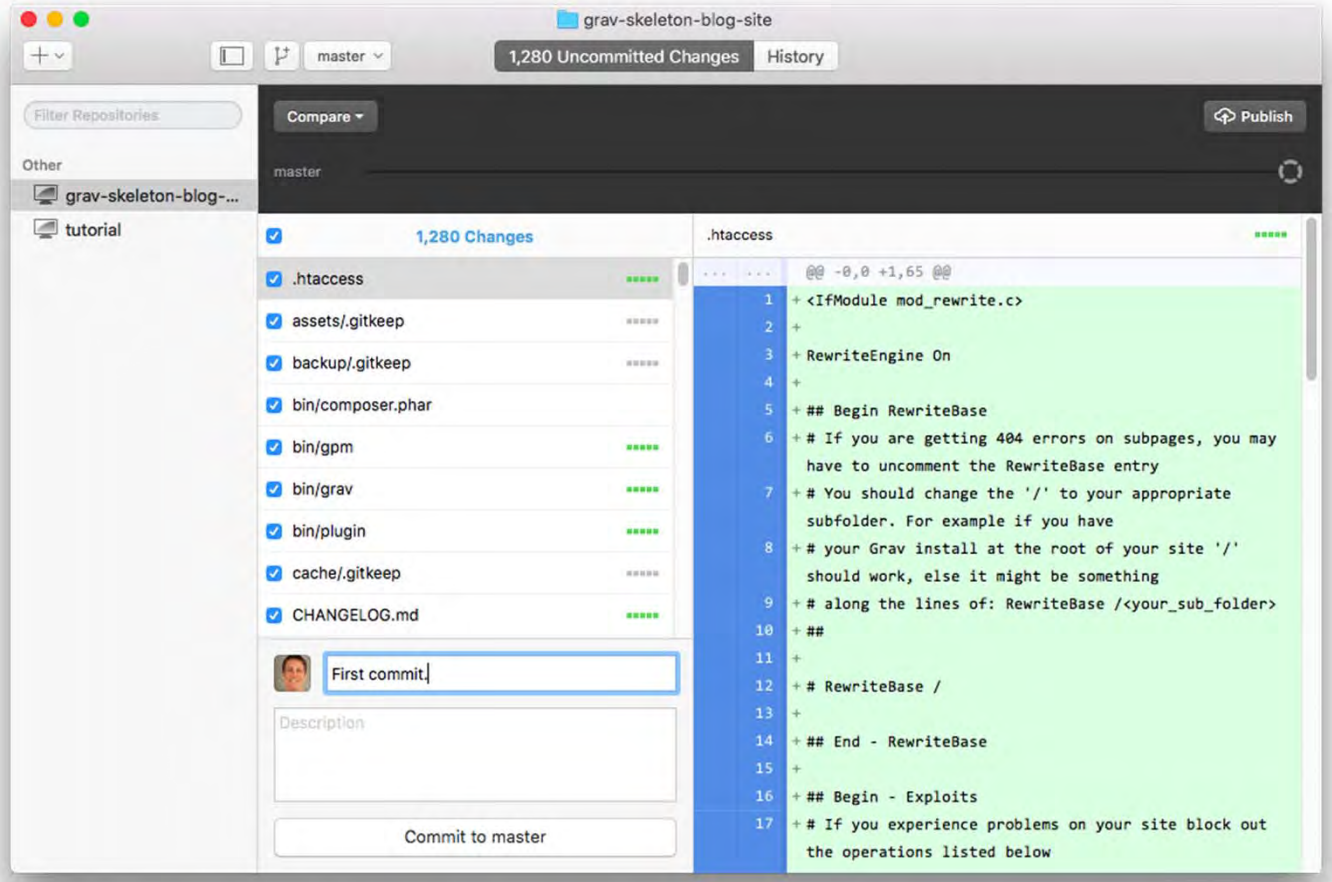
TortoiseGit



https://tortoisegit.org/

GIT – Clientes Gráficos

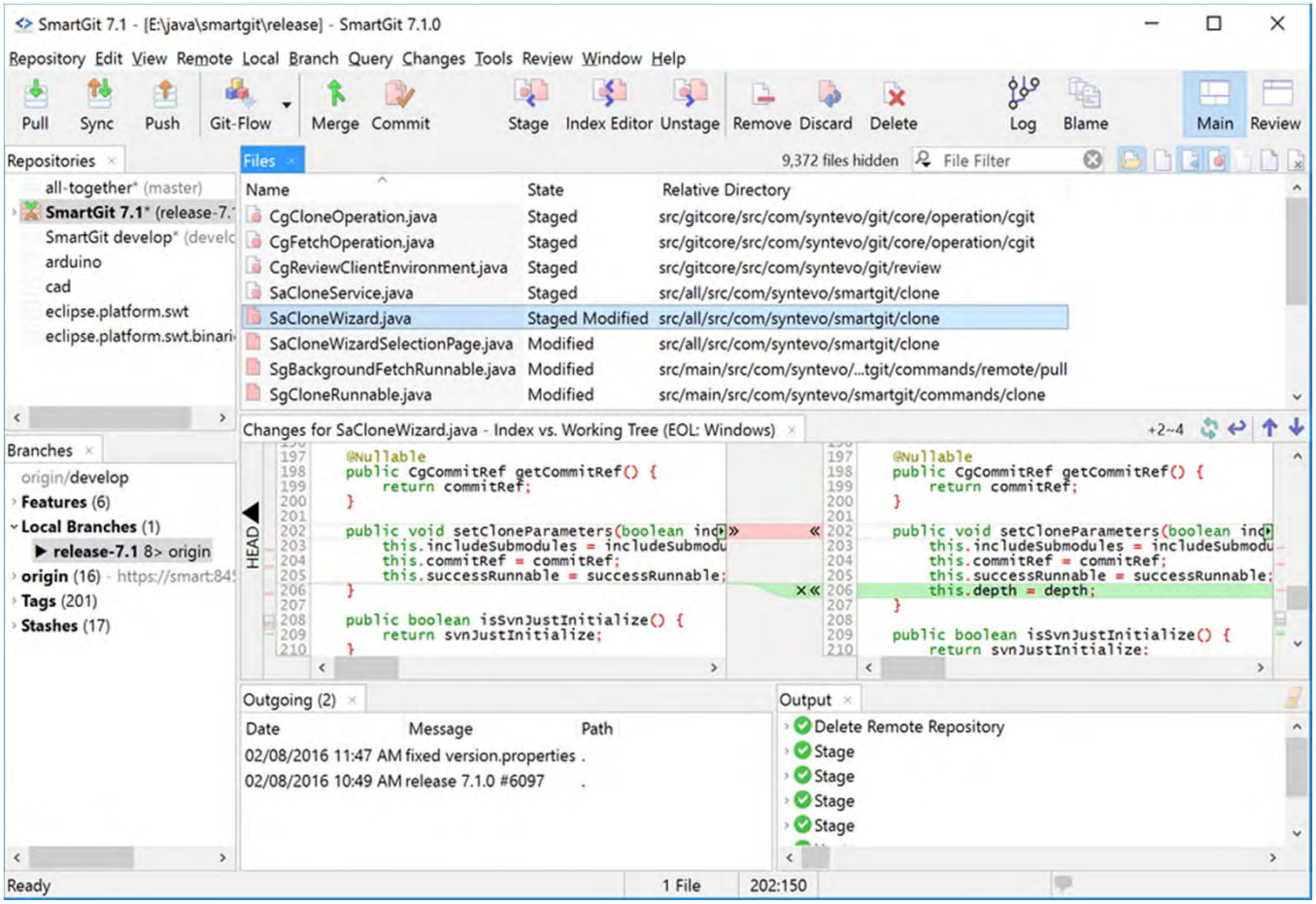
GitHub Desktop



https://desktop.github.com/

GIT – Clientes Gráficos

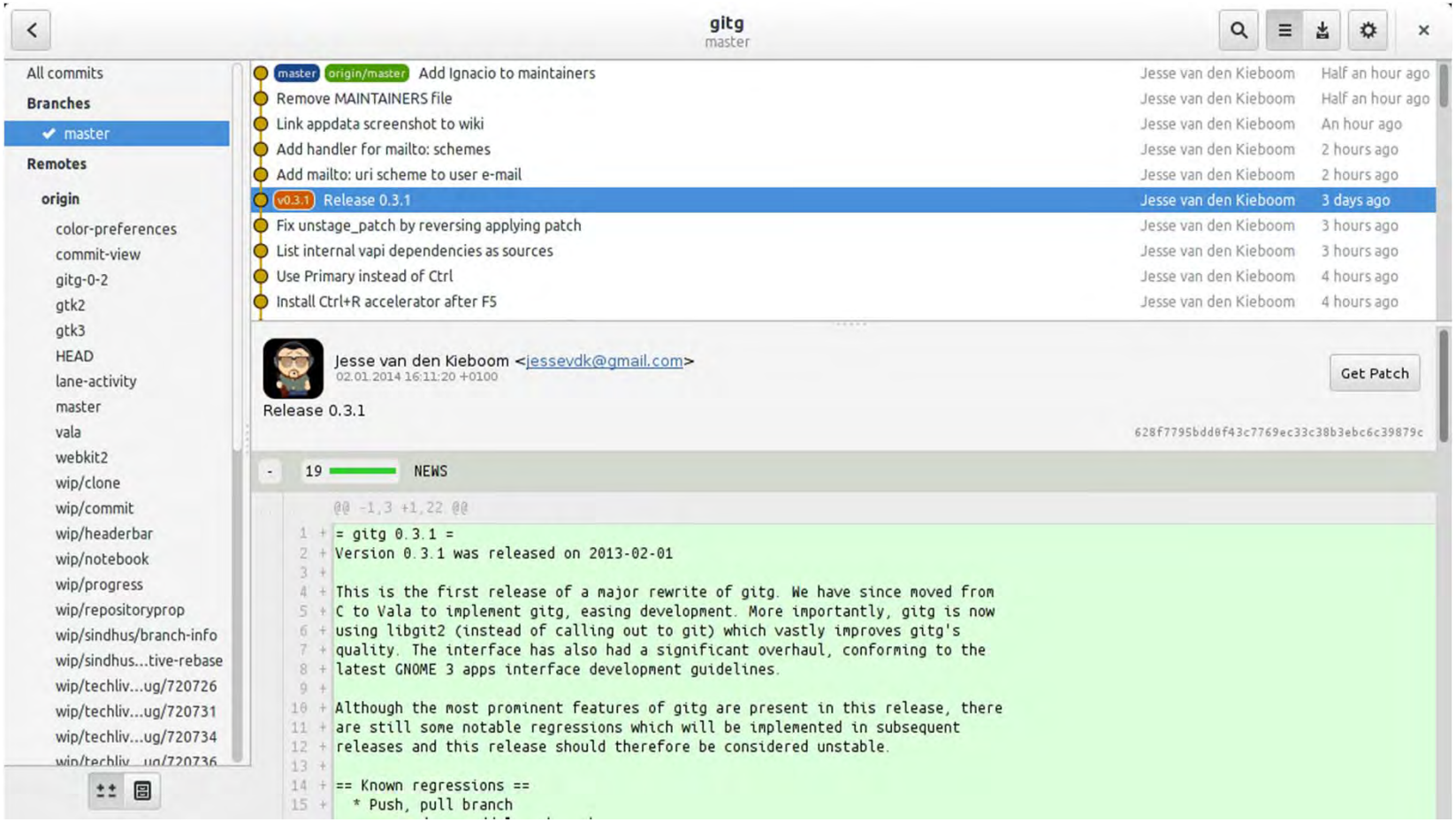
SmartGit



http://www.syntevo.com/smartgit/

GIT – Clientes Gráficos

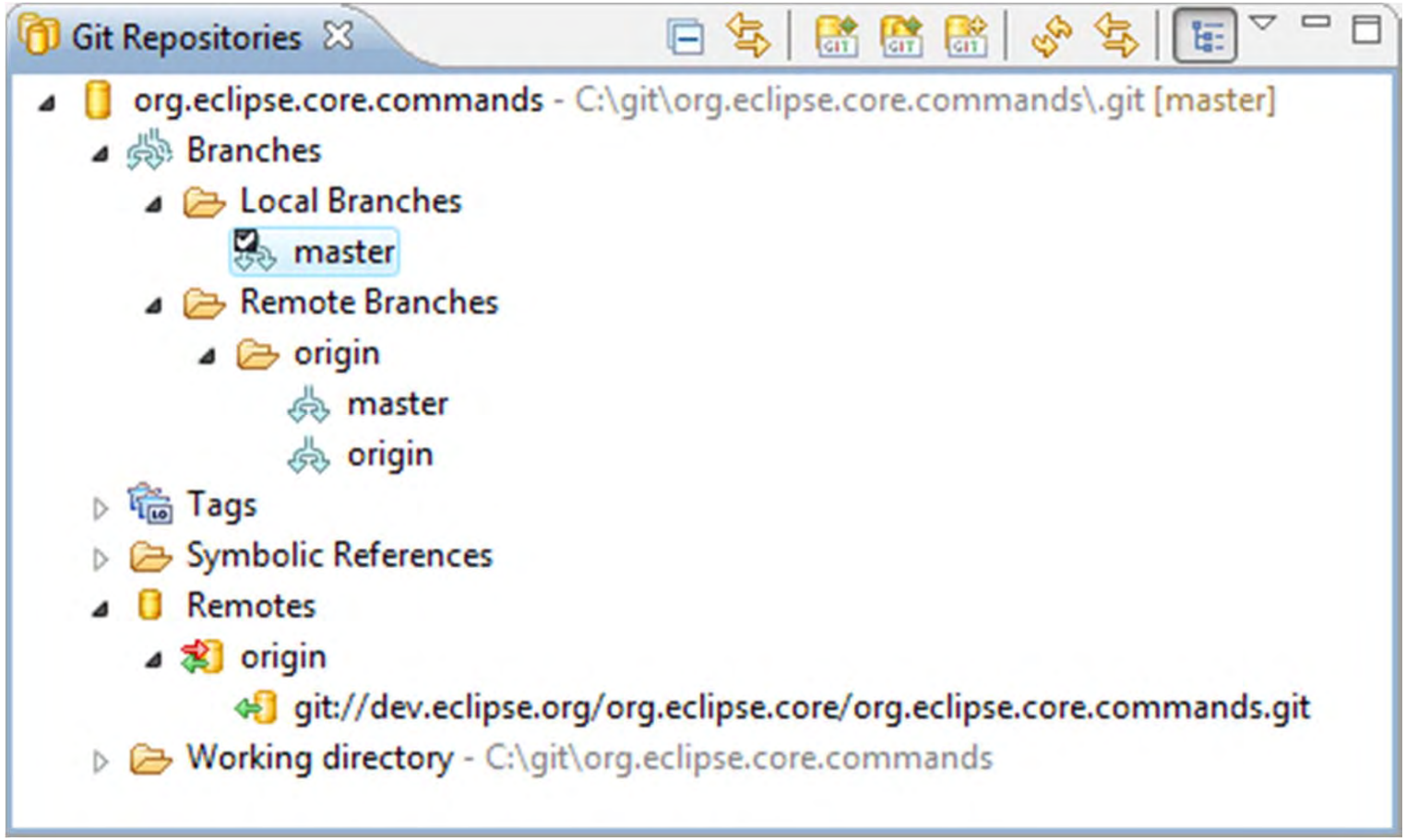
Gitg



https://wiki.gnome.org/Apps/Gitg

GIT – Clientes Gráficos

Egit



http://www.eclipse.org/egit/

git

NETT FORMACIÓN

GitHub

git

NETT FORMACIÓN

https://guides.github.com/activities/hello-world/

git

NETT FORMACIÓN