Universidad de Oviedo

School of Computer Science

EN
English

SOFTWARE ARCHITECTURE

Software Architecture

Introduction to GIT

2015

Jose Emilio Labra Gayo

Version Control Systems

- Centralized
  - A central repository contains all the code
  - Examples: CVS, Subversion,...
- Distributed
  - Each user has its own repository
  - Examples: mercurial, git, ...

Git

Designed by Linus Torvalds (Linux), 2005

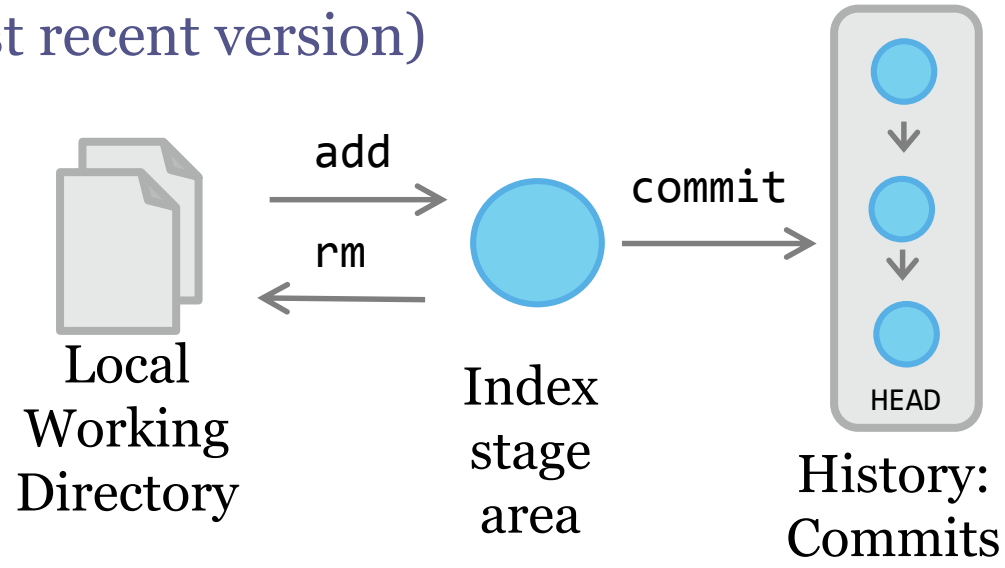
Goals:

- Applications with large number of code files
  - Efficiency
- Distributed teamwork
  - Each developer has his own repository
    - Local copy of all the change history
    - It is possible to do commits even without internet connection
- Support for Non-linean development (branchs)

Local components

3 main local components:

- Local working directory
- Index: stage area, sometimes also called cache
- History: Stores versions or commits
  - HEAD (most recent version)



Branching

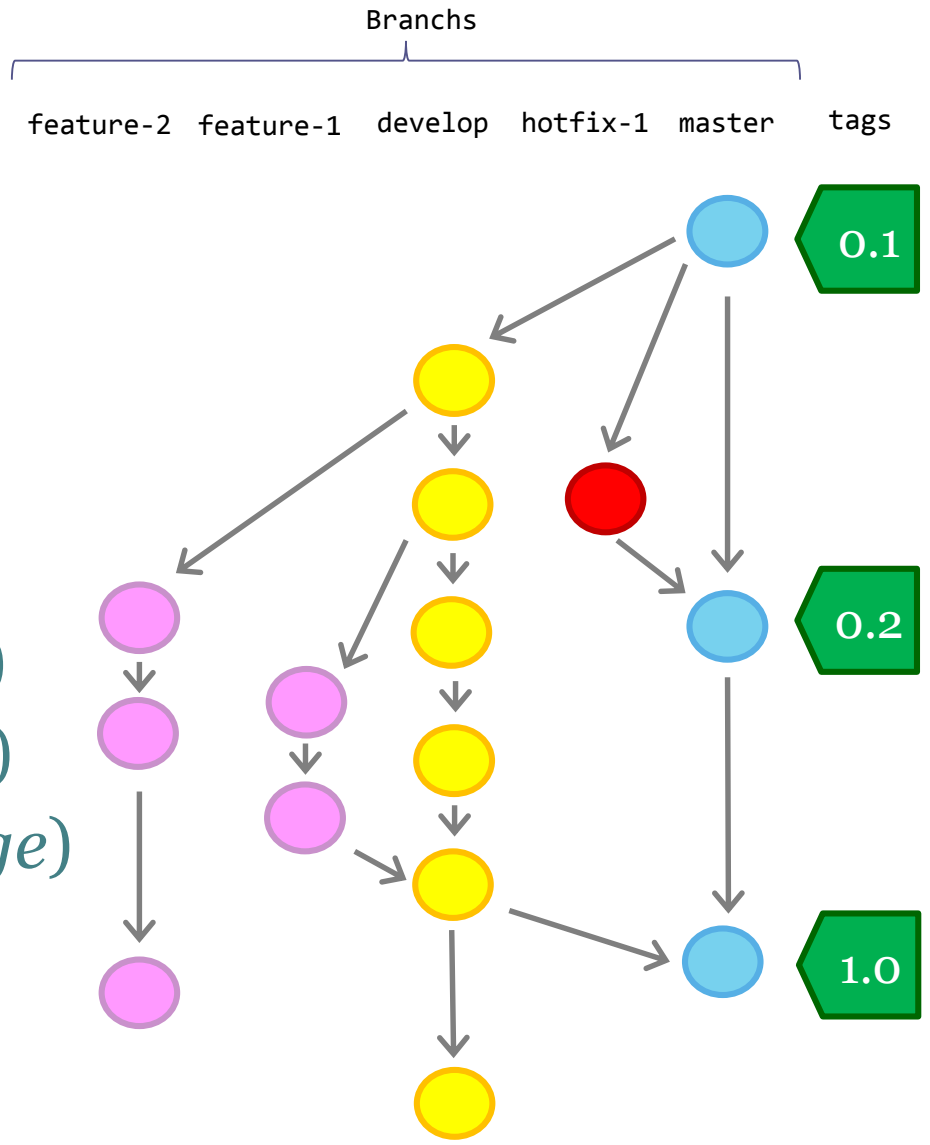
Git facilitates branch management

- master = initial branch

Operations:

- Create a branch (*branch*)
- Switch branch (*checkout*)
- Combine branches (*merge*)
- Tag branches (*tag*)

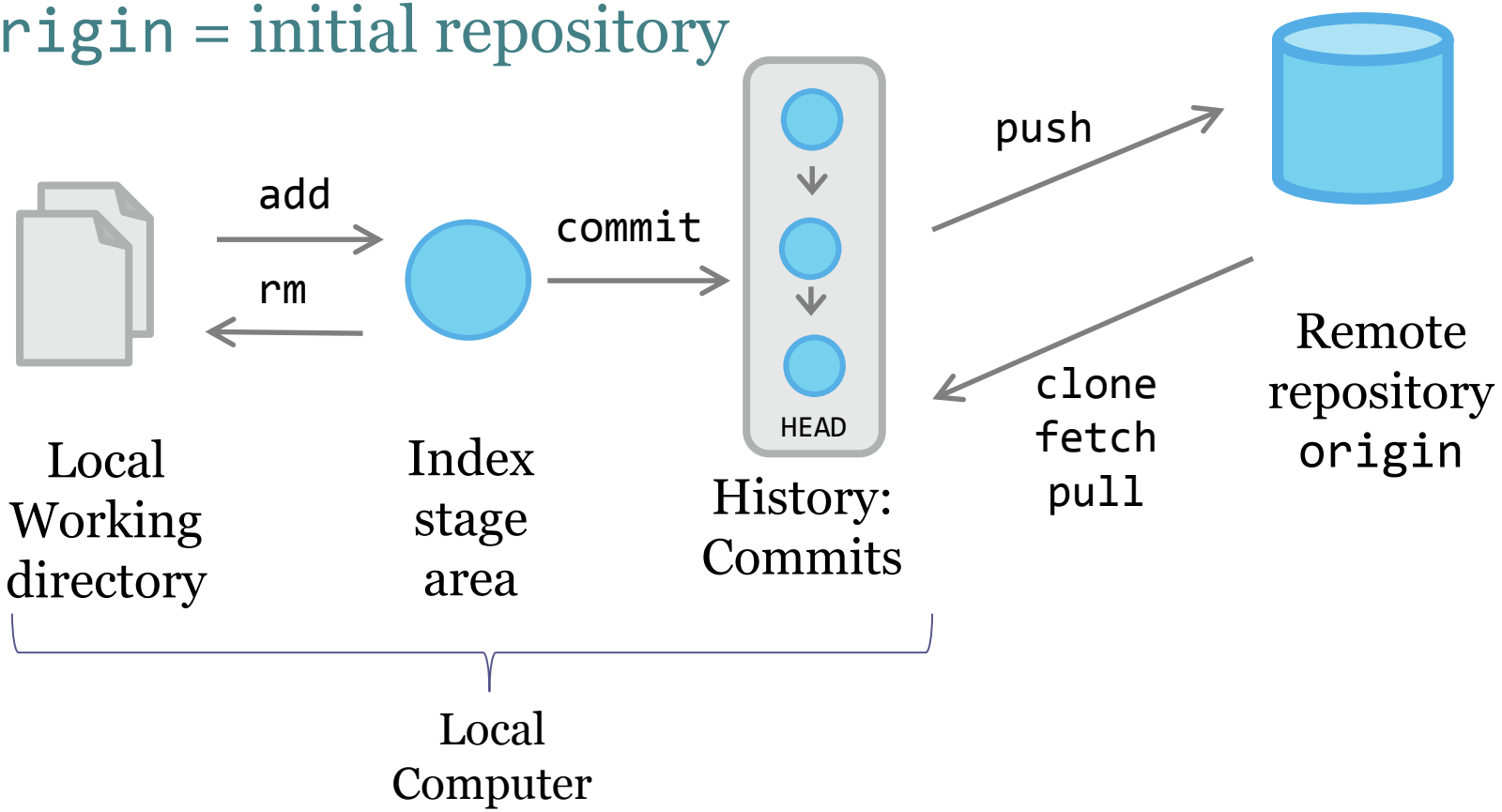
Several branching styles



Remote repositories

It is possible to connect with remote repositories

`origin` = initial repository



Basic usage

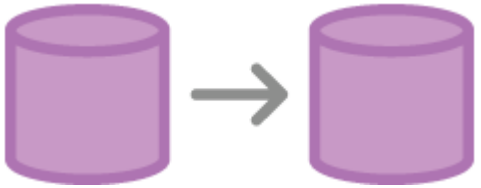
```
init
clone
config
add
commit
status
log
diff
```

init - Create repositories

`git init`

- Transforms current folder in a *git* repository
- Creates folder .git
- Variants:
  - `git init <folder>`
    - Creates an empty repository in a given folder
  - `git init --bare <folder>`
    - Initializes a Git repository but omits working directory

NOTE: This is instruction is usually done once



clone - clone repositories

`git clone <repo>`

Clones repository <repo> in the local machine

<repo> can be in a remote machine

Example:

```
git clone https://github.com/Arquisoft/Trivial0.git
```

NOTA: Like `init`, this instruction is usually done once

config - Configure git

```
git config --global user.name <name>
```

- Declares user name
- Other configuration options:
  - user.email, merge.tool, core.editor, ...
- Configuration files:
  - <repo>/.git/config -- Repository specific
  - ~/.git/config -- Global

add - Add to the index

```
git add <file>
git add <dir>
```

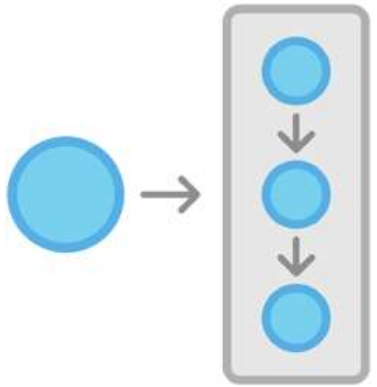
Adds a file or directory to the index

Variants

`git add --all` = Add/delete files

Index or stage area stores file copies before they are included in the history

commit - Add to the history



```
git commit
git commit -m "message"
```

- Add files from index to history
  - Creates a new project snapshot
  - Each snapshot has an SHA1 identifier
    - It is possible to recover it later
    - It is possible to assign tags to facilitate management

NOTE: it is convenient to exclude some files from control version
Examples: binaries (*.class), temporals, configuration (.settings), private (Database keys...), etc.
`.gitignore` file contains the files or directories that will be excluded

status - info about index

```
git status
```

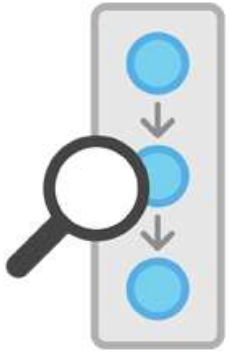
Shows *staged*, *unstaged* and *untracked* files

staged = in index but not in history

unstaged = modified but not added to index

untracked = in local working directory

log - info about history



`git log`

Shows changes history

Variants

`git log --oneline`	1 line overview
`git log --stat`	Statistics
`git log -p`	Complete path with diff
`git log --autor="expr"`	Commits of one author
`git log --grep="expr"`	Searches commits

`git log --graph --decorate --online`

Shows changes graph

diff - Shows differences

`git diff`

Working directory vs index

`git diff --cached`

Index vs Commit

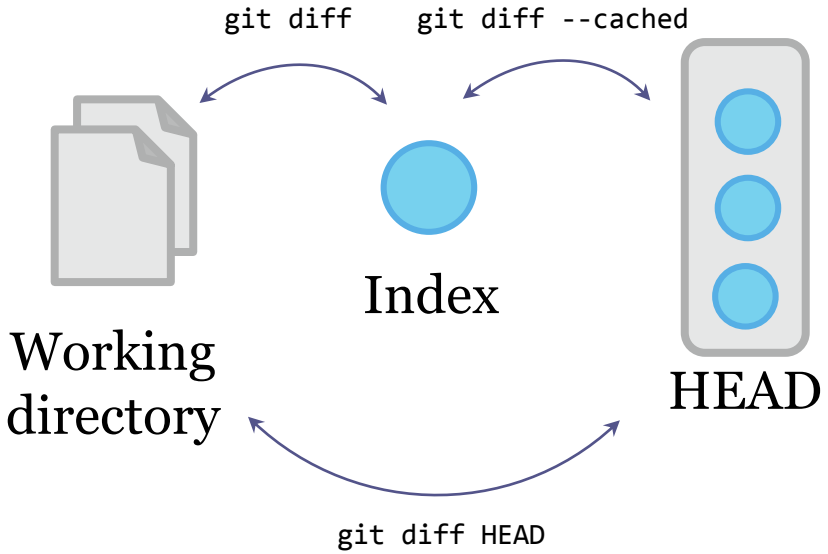
`git diff HEAD`

Working directory vs commit

Some options:

--color-words

--stat



Undoing changes

Commands to undo changes

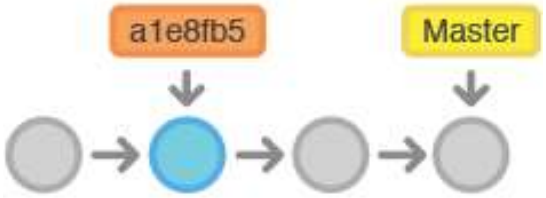
- checkout
- revert
- reset
- clean

checkout

Changes working directory

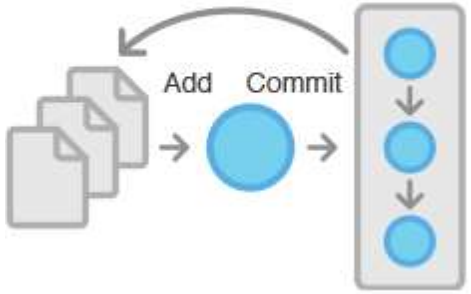
`git checkout <c>` Change to commit `<c>`

It changes to state "*detached HEAD*"



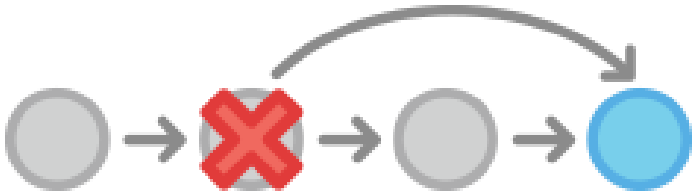
`git checkout <c> <f>`

Recovers file `<f>` from commit `<c>`



NOTE:
`checkout` can also used to change to different branches

revert

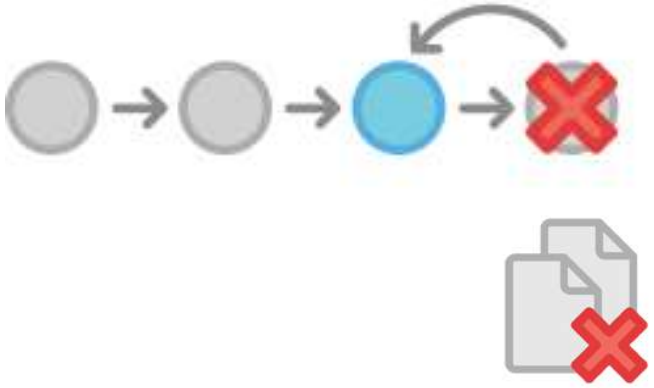


`git revert <c>` Recovers commit `<c>`

Adds the recovered version to the history

Note: Safe operation
It is possible to track changes in the history

reset



Undo changes

Unsafe operation

`git reset`	Undo index changes
`git reset --hard`	Undo index and working directory changes
`git reset <c>`	Undo changes and recover commit `<c>`

NOTE: It can be dangerous to do reset in repositories that have been published. It is better to use `revert`

clean

Deletes local files

`git clean -f` Deletes *untracked* files

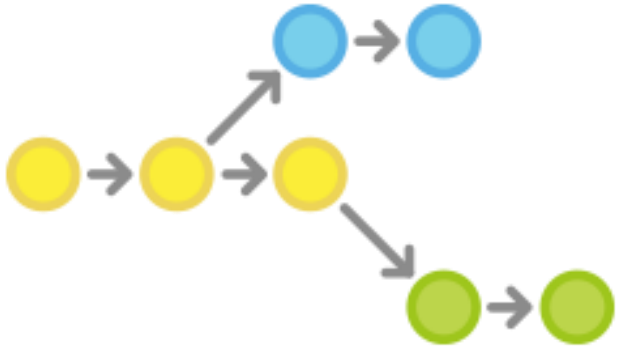
NOTE: Unsafe (it is possible to lose local changes)

`git clean -n` Shows which files would be deleted

Branching

```
branch
checkout
merge
```

branch



Branch management

`git branch` Shows existing branches

`git branch <r>` Creates branch <r>

`git branch -d <r>` Delete branch <r>

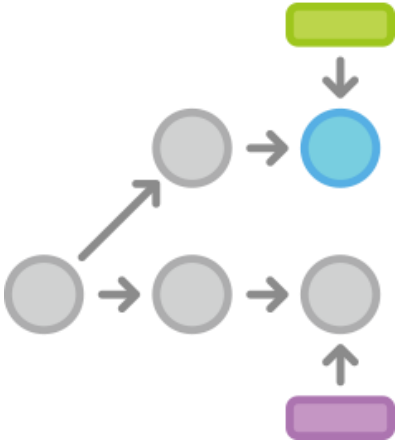
Safe (it doesn't delete if there are pending merges)

`git branch -D <r>` Delete branch <r>

Unsafe (deletes a branch and its commits)

`git branch -m <r>` Rename current branch to <r>

checkout



Change local directory to a branch

`git checkout <r>`

Changes to existing branch `<r>`

`git checkout master` Changes to branch master

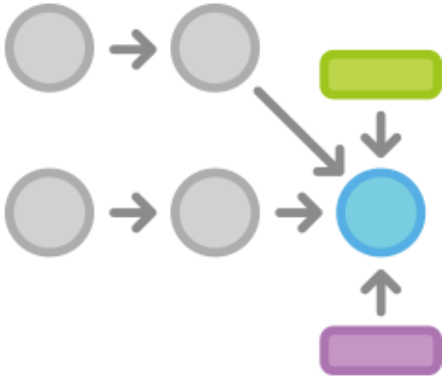
`git checkout -b <r>`

Creates branch `<r>` and changes to it

Equivalent to

```
git branch <r>
git checkout <r>
```

merge



Combine two branches

`git merge <r>`	Merge current branch with `<r>`
`git merge --no-ff <r>`	Merge generating a commit (safer)

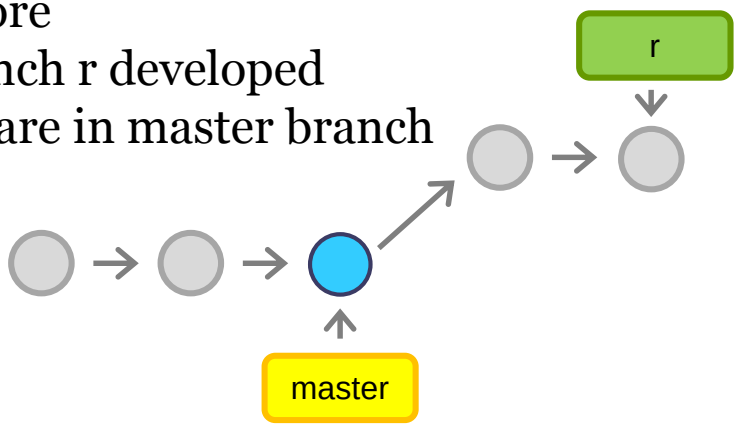
2 merge types:

- Merge fast-forward
- 3-way merge

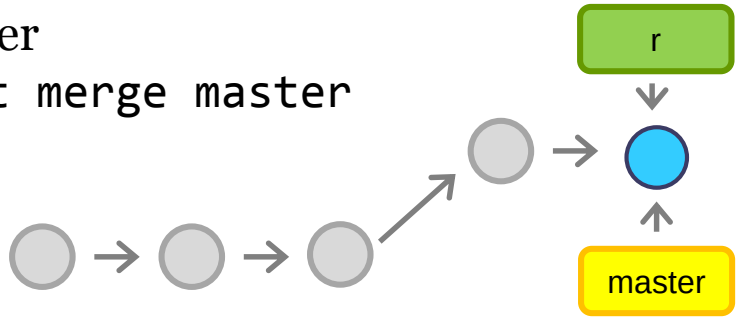
merge fast-forward

When there is lineal path between current branch and the branch to merge

Before
Branch r developed
We are in master branch



After
`git merge master`



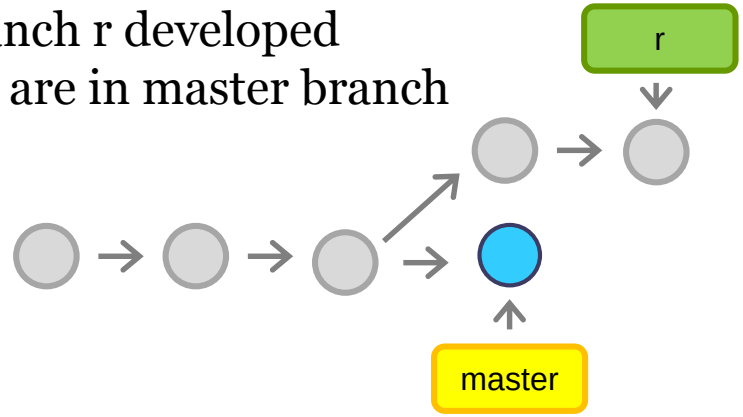
3 way merge

When branches diverge and it is not possible *fast-forward*

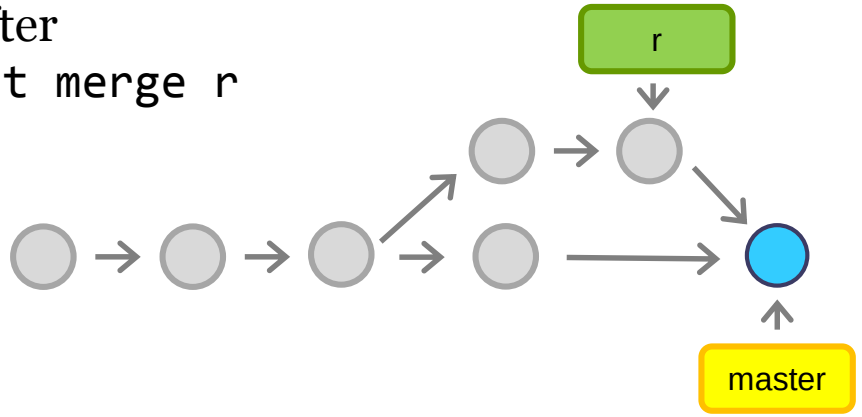
If there are conflicts:

- Show: `git status`
- Edit/repair:
- `git add .`
- `git commit`

Before
Branch r developed
We are in master branch



After
`git merge r`



Remote repositories

```
remote
fetch
pull
push
```

remote - Connect repositories

`git remote`

Show external repositories

`git remote add <n> <uri>`

Create a connection named `<n>` to `<uri>`

`git remote rm <n>`

Remove connection `<n>`

`git rename <before> <new>`

Rename connection `<before>` to `<new>`

NOTAS: `git clone` creates automatically a connection called `origin`
It is possible to have connections to more than one external repository

fetch

Fetch elements from remote repository

- It can download external branches
- Safe operation: does not merge with local files

```
git fetch <remote>
```

- Download branches from <remote>

```
git fetch <remote> <branch>
```

- Download <branch> from repository <remote>

NOTE: Assigns FETCH_HEAD to the head of branch fetched
Branch naming convention: <remote>/<branch>
Example: origin/master

pull

```
git pull <remote>
```

- Fetch from remote repository y merge it
- Equivalent to:

```
git fetch
git merge FETCH_HEAD
```

push

`git push <remote> <branch>`

Send commits from local repository to remote one

Variants

`git push <remote> --all`

Send all branches

If there are changes in remote repository ⇒ Error (non-fast-forward)
Solution:

1. Pull changes and merge with local repositories
2. Push again

NOTE: It is also possible to use option: `--force` (not recommended)

Modify the history

```
commit --amend
rebase
reflog
```

Commit -amend

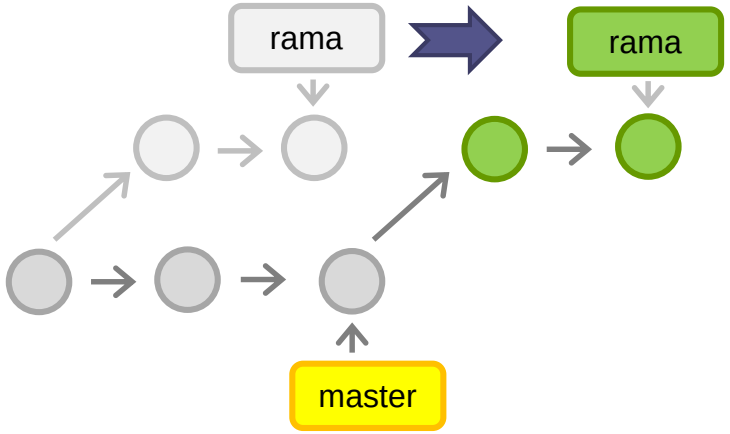
Modify a commit

```
git commit --amend
```

It is possible to add new files from index to the current commit

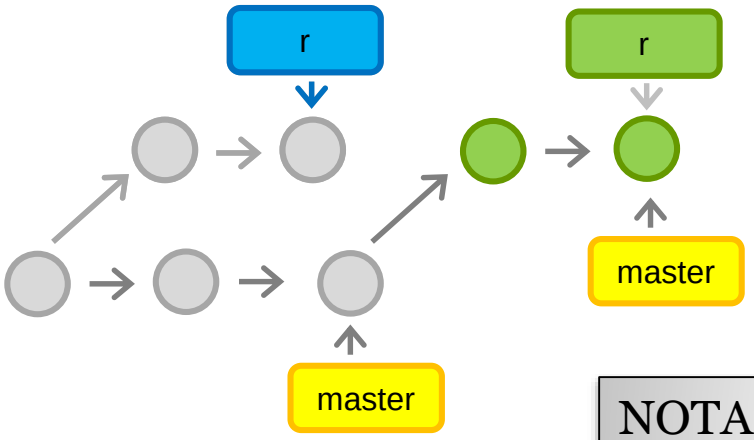
NOTA: It is recommended to avoid this in public commits

rebase
Move a branch



`git rebase <c>` Move current branch to commit `<c>`

Changes in one branch can be based on changes of other branches




```
git checkout r
git rebase master

git checkout master
git merge r
```

NOTA: It is recommended to avoid rebases in commits that have already been published

interactive rebase

`git rebase -i`

Controls which commits are kept in the history

- It can be used to clean the history
- An editor appears with several instructions:
  - `pick` - use the commit as is
  - `reword` - modify commit message
  - `squash` - use commit but hide it
  - ...

reflog - history movements

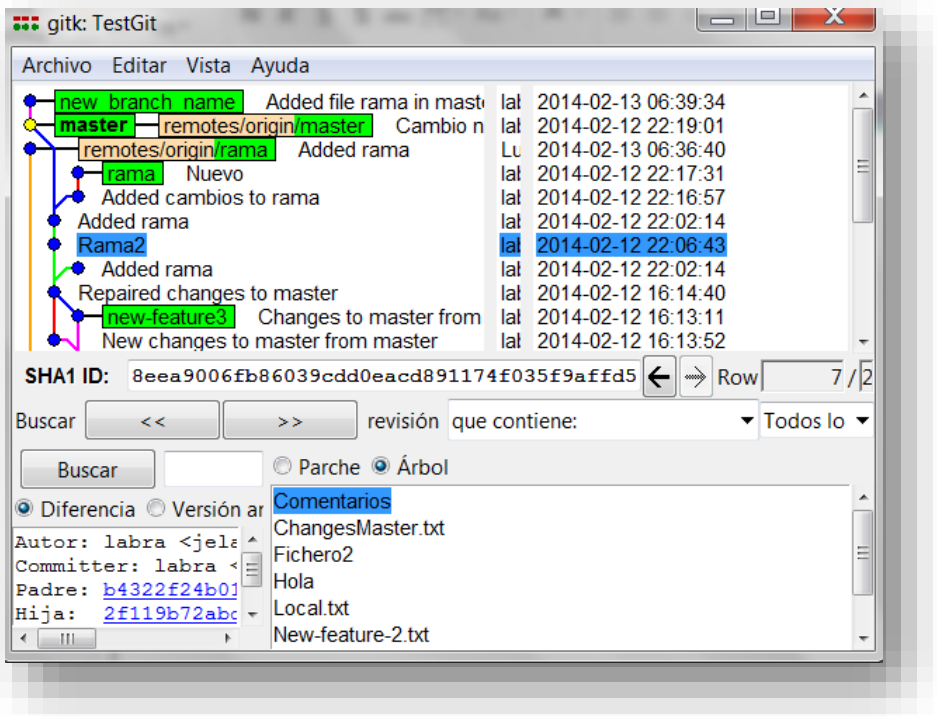
`git reflog`

- Shows history movements
  - git stores information about all the movements that happen in the different branches
    - git reflog shows the changes even if they are not in any branch

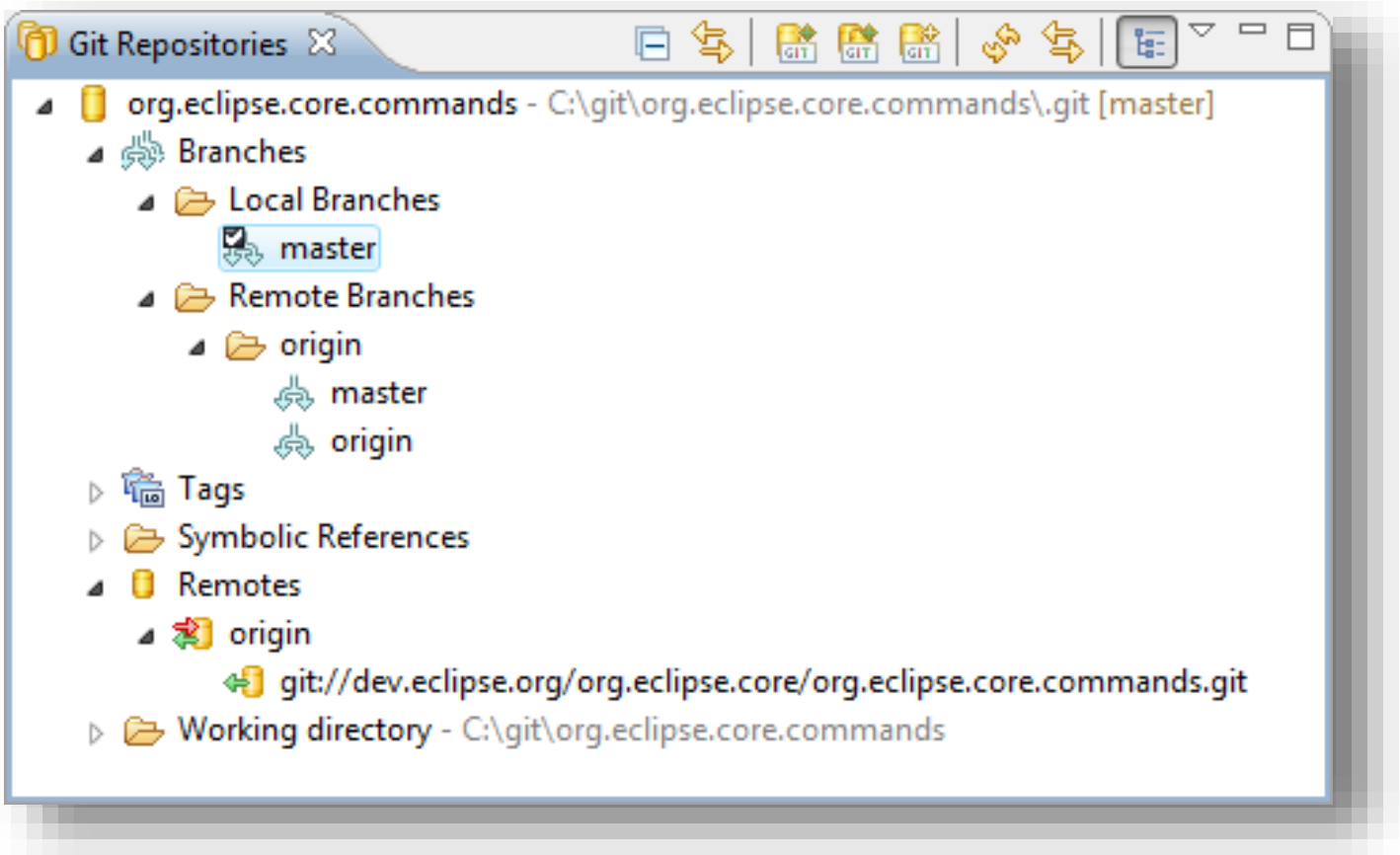
Git tools

git - Command line

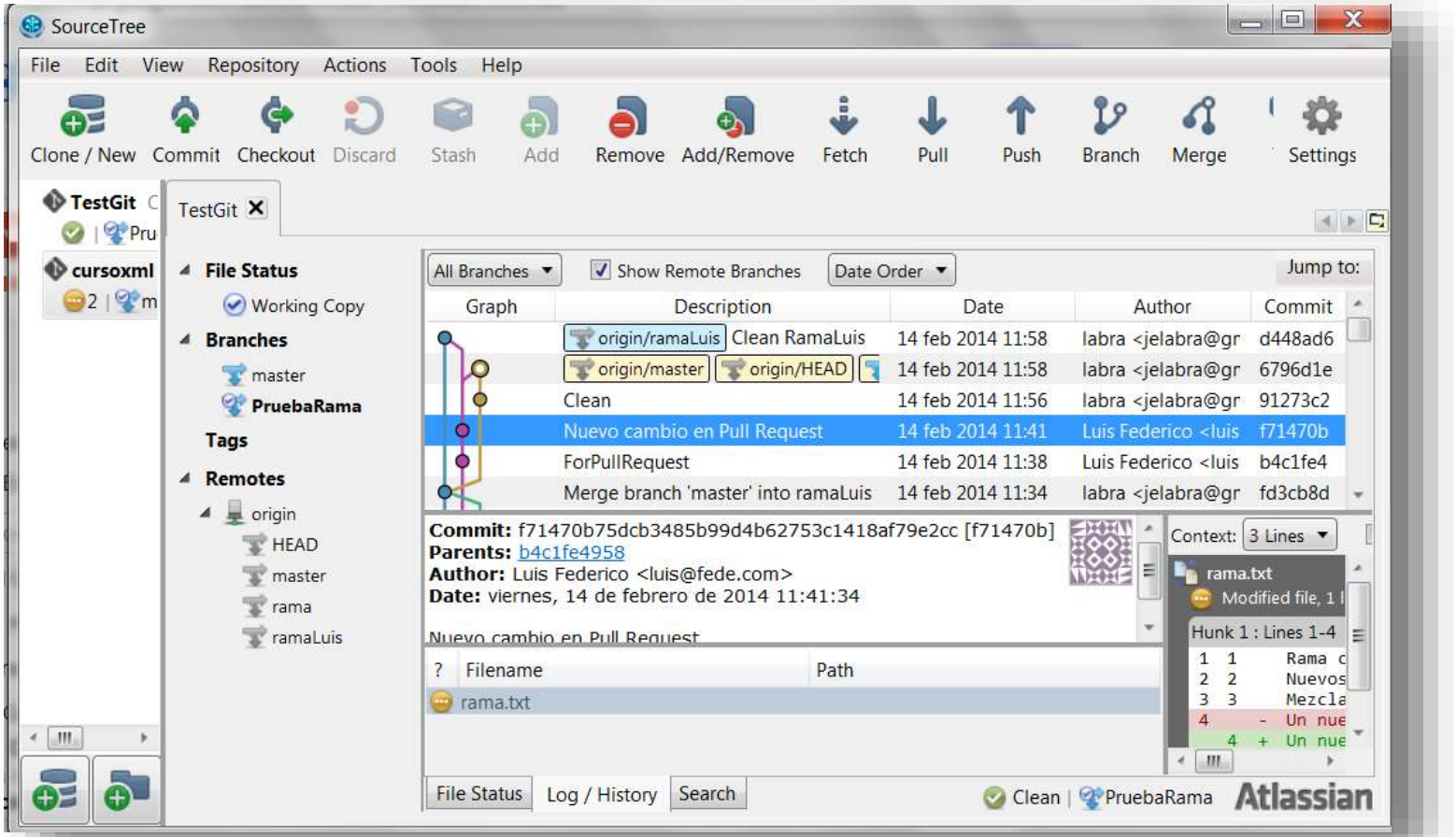
`gitk` Graphical interface



EGit - Eclipse environment

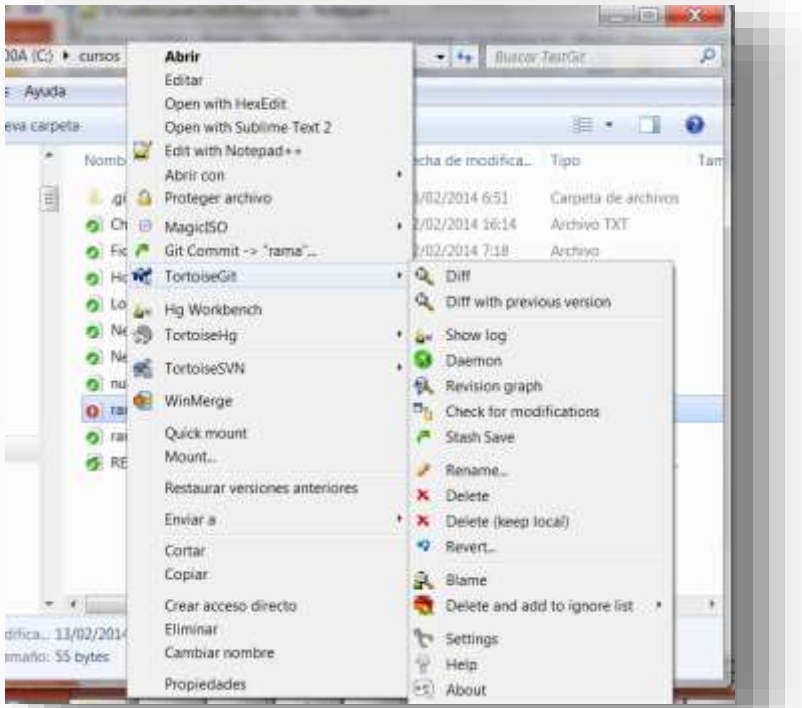


SourceTree


SourceTree
File Edit View Repository Actions Tools Help
Clone / New
Commit
Checkout
Discard
Stash
Add
Remove
Add/Remove
Fetch
Pull
Push
Branch
Merge
Settings
TestGit
cursoxml
File Status
Working Copy
Branches
master
PruebaRama
Tags
Remotes
origin
HEAD
master
rama
ramaLuis
All Branches
Show Remote Branches
Date Order
Jump to:
Graph
Description
Date
Author
Commit
origin/ramaLuis Clean RamaLuis 14 feb 2014 11:58 labra <jelabra@gr d448ad6
origin/master origin/HEAD 14 feb 2014 11:58 labra <jelabra@gr 6796d1e
Clean 14 feb 2014 11:56 labra <jelabra@gr 91273c2
Nuevo cambio en Pull Request 14 feb 2014 11:41 Luis Federico <luis f71470b
ForPullRequest 14 feb 2014 11:38 Luis Federico <luis b4c1fe4
Merge branch 'master' into ramaLuis 14 feb 2014 11:34 labra <jelabra@gr fd3cb8d
Commit: f71470b75dcb3485b99d4b62753c1418af79e2cc [f71470b]
Parents: b4c1fe4958
Author: Luis Federico <luis@fede.com>
Date: viernes, 14 de febrero de 2014 11:41:34
Nuevo cambio en Pull Request
Context: 3 Lines
rama.txt
Modified file, 1
Hunk 1 : Lines 1-4
? Filename Path
rama.txt
File Status
Log / History
Search
Clean | PruebaRama
Atlassian

Tortoise Git

Integrates with Windows Explorer



Other tools

http://git-scm.com/downloads/guis

Diffs and merges Management

- p4merge
- kdiff

Github

- Social coding tool
  - Github Inc. company created in 2008
    - 2013: >3 mill. of users, >5mill. projects
- Free management of open source codes
- Public and free projects by default
  - It is possible to have private repositories
  - Student accounts

Github

- Project repository
  - Issues/milestones management
  - Wiki
  - Following repositories, users, etc.
- Pull Requests
  - Request merges and combinations
  - Code reviews
    - It is possible to include comments, see differences, etc.
  - *Pull requests* List management

References

Tutorials

http://rogerdudler.github.com/git-guide/

https://www.atlassian.com/git

http://training.github.com/materials/slides/

http://marklodato.github.io/visual-git-guide/

http://nvie.com/posts/a-successful-git-branching-model/

Videos

http://vimeo.com/49444883