

# Unit: II

## git

### lab session.

## Herramientas Avanzadas para el Desarrollo de Aplicaciones

Languages and computing systems  
University of Alicante

Curso 2014-2015 , Copyleft © 2011-2015 .  
Reproducción permitida bajo los términos de la licencia de  
documentación libre GNU.

# Content

- 1 First steps with git
- 2 Modification of files (I)
- 3 Modification of files (II)
- 4 Adding files
- 5 Repository Log
- 6 Deleting files
- 7 Undo actions(I)
- 8 Undo actions (II)
- 9 Renaming files
- 10 Branches: Current, create, switch branch, differences
- 11 Branches: Differences regarding the commits
- 12 Exercise with branches
- 13 Branches: merge, rebase
- 14 Cloning repositories
- 15 Objectives...
- 16 Delivering...

# First steps with git

```
1  # Check the installed version
   git --version
3
   # We start using the code of the previous session (
     guess a number)
5  # We create the initial repository to put under
     version control
   # the files of that project.
7  cd Juego; git init

9  # We add to the repository the files of the current
     directory and subdirectories
   git add .
11 # We check the status after adding them
   git status
13 # We confirm that we have added them (commit)
   git commit -m 'Primer commit.' -m "Descripcion
     detallada." -a
```

# Modification of files (I)

```
2      # We modify some file
      gedit main.vala Makefile

4      # We see the differences of the work copy with the
      repository
      git diff

6      # we accept the changes, we can do it in several
      ways:

8      # 1- File to file
10git commit -m "Correccion de errores en el prog. ppal.
      " main.vala
      git commit -m "Anyadido objetivo de depuracion."
      Makefile
```

# Modification of files (II)

```
1  # 2- We prepare the 'stage' that will conform the
    commit
    # A
3  git add Makefile
   git commit -m "Modificado Makefile."
5  # B
   git add Makefile main.vala
7  git commit -m "Un solo commit con varios archivos."
```

# Adding files

```
1  # We create the file
   gedit autores.txt
3
   # We add it...
5  git add autores.txt

7  # We confirm the addition...
   git commit -m "fichero autores.txt" autores.txt
9

   # Which files are under version control?
11 git ls-files
```

# Repository Log

```
1 # Which operations have we done over the repository?
2 git log
3
4 commit dbafcc771eb504a541db32d7a7b5287a470516aa # HEAD
5 Author: hada <hada@dlsi.ua.es>
6 Date: Fri Jan 20 18:26:22 2012 +0100
7
8 fichero autores.txt
9
10 commit 18442c5369b1f6f7920a4ebea39ec38bf9b62555 # HEAD~1
11 Author: hada <hada@dlsi.ua.es>
12 Date: Fri Jan 20 17:54:48 2012 +0100
13
14 Cambios al makefile.
15
16 commit e3911a1778c405ecce14bae1c1a97ec81832242b # HEAD~2
17 Author: hada <hada@dlsi.ua.es>
18 Date: Fri Jan 20 17:54:12 2012 +0100
19
20 # Each commit has its own SHA-1 number
21 # SHA-1 numbers can be summarized until they are clear
22 # We can refer to the last commit done (the first of the list) with the alias:
23 HEAD
24
25 # We tag HEAD with v1.0 and HEAD~1 with v0.9
26 git tag v1.0
27 git tag v0.9 HEAD~1
28 git tag -l
29 # We can graphically visualize it with gitk
30 gitk
```

# Deleting files

```
1  # We delete the file
   git rm authors.txt
3
   # We check the repository status
5  git status

7  # We confirm the deletion...
   git commit -m "fichero autores.txt borrado"
9
   # What does the repository log say?
11 git log

13 commit 5a369acdefc3c1d28c7d1c9561f7bb26d9daead3
   Author: hada <hada@dlsi.ua.es>
15 Date:   Fri Jan 20 18:41:56 2012 +0100

17 Archivo autores.txt borrado.
```



# Undo actions (I)

```
2  # We delete the file
   git rm autores.txt
   git commit ...
4
   # NOOO!!! it is a mistake!!!, can I get it back?
6
   # What does the log of the repository says?
8   git log

10  commit 5a369acdefc3c1d28c7d1c9561f7bb26d9daead3 # HEAD
   Author: hada <hada@dlsi.ua.es>
12  Date:   Fri Jan 20 18:41:56 2012 +0100

14  Archivo autores.txt borrado.

16  commit dbafcc771eb504a541db32d7a7b5287a470516aa # HEAD~1
   Author: hada <hada@dlsi.ua.es>
18  Date:   Fri Jan 20 18:26:22 2012 +0100

20  Correccion de errores.
```

# Undo actions (II)

There are several ways of doing it, we will use 'git revert':

```
1  # revert undoes a commit by creating an 'inverse' commit
   # using the option '-n' it does everything except creating the 'inverse'
   commit
3  # we do it this way to see step by step the repository status
   git revert -n HEAD
5
   git status
7
   # On branch master
9   # Changes to be committed:
   #   (use "git reset HEAD <file >..." to unstage)
11  #
   #       new file:   autores.txt
13  #
15  # What does the repository log STILL says?
   git log
17
   commit 5a369acdefc3c1d28c7d1c9561f7bb26d9daead3    # HEAD
19   Author: hada <hada@dlsi.ua.es>
   Date:   Fri Jan 20 18:41:56 2012 +0100
21
   Archivo autores.txt borrado.
23
   # QUESTION: What should we do now?
25
   # Another way of doing it: git reset
27  # Look for it and see what it does and try to use it to solve this situation.
```

# Renaming files

```
1  git mv autores.txt AUTHORS
   git status
3  # On branch master
   # Changes to be committed:
5  #   (use "git reset HEAD <file >..." to unstage)
   #
7  #       renamed:    autores.txt -> AUTHORS
   #
9
11 # QUESTION: What would we have to do now?
13 # Only one commit with the rename operation
   git commit -m "Renombrado archivo autores.txt a AUTHORS."
```

# Branches: Current, create, switch branch, differences

```
1  # Check the current, remote or all
   git branch [-r] [-a]
3  * master

5  # Create a branch called 'devel' based on the current one
   git branch devel
7  # Existing branches, in the one we are there is a '*'
   git branch
9  devel
   * master
11 # Switch to 'devel'
   git checkout devel
13 Switched to branch 'devel'

15 # Check that you are in the branch 'devel'
   # Do modifications to AUTHORS here and save them (commit)
17 # switch again to 'master'
   git checkout master
19 Switched to branch 'master'

21 # Differences between 'devel' and 'master' (we are in master)
   git diff devel
23 # or also
   git diff master devel
25 git diff devel master
```

# Branches: Differences regarding the commits

We do it using 'git log [-p]'

```
1  # We can see it in several ways
   git log master..devel # source branch: master, target branch: devel
3  git log devel..master
   git log master..      # source branch: master, target branch: current
5  git log ..master      # source branch: current, target branch: master

7  # we can also use show-branch, e.g.:
   git show-branch master devel
9  ! [master] Renombrado archivo.
   * [devel] Cambios en README.
11  —
   * [devel] Cambios en README.
13  +* [master] Renombrado archivo.
```

# Exercise with branches

- Now you have already created the branch 'devel'.
- You should be working in that one and not in 'master'.
- Modify the needed files so in the game we allow a maximum number of attempts for guessing the number, for instance 3.
- When done, you should have all the modifications in the 'devel' branch (you should have done the needed commits)

# Branches: merge, rebase

- Let's suppose we want to change the modifications of the 'devel' branch to 'master'.
- We have two options: **merge** them or **rebase** them

```
1  # Merge option
   git merge devel
3  Updating ddeebdc..646eba9
   Fast-forward
5  README |      1 +
   1 files changed, 1 insertions(+), 0 deletions(-)
7
   # Rebase option
9  git rebase devel
   First, rewinding head to replay your work on top of it...
11 Fast-forwarded master to devel.
```

# Clonning repositories

- It allows us 'copying' a local repository or in a remote way (http/s, ssh, git, git+ssh).
- A link between them is stablished which allows doing **pull**, **fetch**, **merge** and **push** operations.

```
1  # clone the repository of the code of the practical assignment 1
   # we do it in a directory called practical1b
3  git clone practical practical1b
   Cloning into 'practical1b'...
5  done.

7  cd practical1b; ls  # it should be everything: work copy + repository (.git)

9  # If we try to do push we will obtain an error. The source repository
   # contains a working copy (it is not bare —only .git directory—)
```

Clonning repositories is very useful when we clone a repository with only data and is not a working copy (bare).



# Objectives...

The students knows how:

- ☐ to create a repository using git and add the files that will be under the version control system.
- ☐ to do commits of the actions done (files modifications, adding new files, deleting files, renaming files, etc. . . )
- ☐ to do the 'log' of the actions performed.
- ☐ to tag certain version of the files.
- ☐ to undo actions, e.g. recover a deleted file.
- ☐ to create branches, switch branches, check differences between branches.
- ☐ to import changes from a branch to another.
- ☐ to clone repositories.

- What you have to deliver in this assignment is the working directory (Juego) compressed in a file called `juego.tgz`.
- Remember that this directory already contains the working copy and the `'.git` directory'.