Unit: II
git
lab session.

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Languages and computing systems
University of Alicante

Content

First steps with git

```
# Check the installed version
git --version

# We start using the code of the previous session (
    guess a number)
# We create the initial repository to put under
    version control
# the files of that project.
cd Juego; git init

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

Modification of files (I)

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

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

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

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

Modification of files (II)

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

Adding files

```
# We create the file
gedit autores.txt

# We add it...
git add autores.txt

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

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

Repository Log

```
# Which operations have we done over the repository?
git log

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

fichero autores.txt

commit 18442c5369b1f6f7920a4ebea39ec38bf9b62555    # HEAD~1
Author: hada <hada@dlsi.ua.es>
Date:   Fri Jan 20 17:54:48 2012 +0100

Cambios al makefile.

commit e3911a1778c405ecce14bae1c1a97ec81832242b    # HEAD~2
Author: hada <hada@dlsi.ua.es>
Date:   Fri Jan 20 17:54:12 2012 +0100

# Each commit has its own SHA-1 number
# SHA-1 numbers can be summarized until they are clear
# We can refer to the last commit done (the first of the list) with the alias:
     HEAD

# We tag HEAD with v1.0 and HEAD~1 wih v0.9
git tag v1.0
git tag v0.9 HEAD~1
git tag -l
# We can graphically visualize it with gitk
gitk
```

Deleting files

```
# We delete the file
git rm authors.txt

# We check the repository status
git status

# We confirm the deletion...
git commit -m "fichero autores.txt borrado"

# What does the repository log say?
git log

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

Archivo autores.txt borrado.
```

Undo actions (I)

```
# We delete the file
git rm autores.txt
git commit ...

# NOOO!!! it is a mistake!!!, can I get it back?

# What does the log of the repository says?
git log

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

 Archivo autores.txt borrado.

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

   Correccion de errores.
```

Undo actions (II)

There are several ways of doing it, we wil use ‘`git revert`’:

```
# revert undoes a commit by creating an 'inverse' commit
# using the option '-n' it does everything except creating the 'inverse'
    commit
# we do it this way to see step by step the repository status
git revert -n HEAD

git status

# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   autores.txt
#

# What does the repository log STILL says?
git log

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

Archivo autores.txt borrado.

# QUESTION: What should we do now?

# Another way of doing it: git reset
# Look for it and see what it does and try to use it to solve this situation.
```

Renaming files

```
git mv autores.txt AUTHORS
git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       renamed:    autores.txt -> AUTHORS


# QUESTION: What would we have to do now?

# Only one commit with the rename operation
git commit -m "Renombrado archivo autores.txt a AUTHORS."
```

Branches: Current, create, switch branch, differences

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

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

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

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

Branches: Differences regarding the commits

We do it using ‘`git log [-p]`’

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

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

```
# Merge option
git merge devel
Updating ddeebdc..646eba9
Fast-forward
 README |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)

# Rebase option
git rebase devel
First, rewinding head to replay your work on top of it...
Fast-forwarded master to devel.
```

Cloning 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.

```
# clone the repository of the code of the practical assignment 1
# we do it in a directory called practica1b
git clone practica1 practica1b
Cloning into 'practica1b'...
done.

cd practica1b; ls  # it should be everything: work copy + repository (.git)

# 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.

Delivering...

- 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'.