# git

Αχιλλέας Πιπινέλης

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Μονάδα Αριστείας ΕΛ/ΛΑΚ ΤΕΙ Αθήνας

# Overview

- Version control in general
- Centralized version control systems
- Distributed version control systems

# Reasons to Use Version Control

- Have a record of what you did, when and why
- Snapshots of different development states
- Compare different versions of files
- Easily share work with others
- Integrate changes from others (merging)
- Mark finished product versions (tags)
- Try out new ideas (branches)

# Glossary

- Repository
  - Stores complete history, branches, tags
- Working Copy
  - Your “playground”, actual source code
- Commit / Revision
  - A single version
- Branch
  - A separate line of development

# Distributed Version Control

- Every user has a full copy of the repository
- Repositories can be synchronized (push/pull)
- Off-line access
- Git, Mercurial, Monotone, Fossil, Bazaar

# Centralized Version Control

- Single repository on a server
- Each user has a working copy
- Only privileged users can commit
- CVS,  Subversion(Facebook - ~2012)

# Git

- Started in 2005 by Linus Torvalds
- Used for Linux kernel development
- Highly distributed
- Cryptographic integrity (SHA-1)
  - http://git-scm.com/book/en/Git-Tools-Revision-Selection#A-SHORT-NOTE-ABOUT-SHA-1
  - http://mikegerwitz.com/papers/git-horror-story

# Install & Setup

- Install
- Set user information

```
# yum install git
$ git config --global user.name 'Jeff Lebowski'
$ git config --global user.email 'thedude@overtheline.com'

$ git config --global core.autocrlf false (windoze)
```

# Creating repositories

- Create a new repository

```
$ cd path/to/project
$ git init
OR
$ git init new_dir
$ cd new_dir
```

# Adding files

- Adding new files

```
$ >foo echo 'bar'
$ git add foo
```

# Writing history

- Permanently save your work

$ git commit

# Updating files

- Tell Git about new changes
- Same thing as adding new files (Git only cares about the content of files)

```
$ >>foo echo 'new content'
$ git add foo
$ git commit -m 'Updated file'
```

# Working Copy Status

- Shows list of modified, new and staged files
- Also reminds you of important commands

```
$ git status
## On branch master
nothing to commit (working directory clean)
```

# Inspecting changes

- Show current changes
- Show differences between commits
- Show differences between branches

```
$ git diff
$ git diff HEAD^ HEAD
$ git diff branch1 branch2
```

# Branching history

- Try out new features
- In an ideal world: one branch per feature
- Branches are cheap, use them often
- Branches can be deleted

```
$ git branch newbranch
$ git checkout newbranch

$ git branch
  master
* newbranch
```

# Merging changes

- Integrate changes from a branch
- Integrate changes from others (more later)

```
$ git merge newbranch
```

# Forking existing projects

- Get a copy of the repository
- fetch/pull to update your copy

```
$ git clone git://git.kernel.org/pub/scm/git/git.git git
$ cd git

$ git fetch
$ git pull
```

# Collaborating

- Use Git's distributed powers
- Pull branches from remote repositories
- Define bookmarks for repositories

$ git pull git://example.com/repo.git feature_branch

OR

$ git remote add repo git://example.com/repo.git

$ git pull repo feature

# Links

- http://git-scm.com
- http://progit.org
- Linus' infamous Tech-talk at Google: https://www.youtube.com/watch?v=4XpnKHJAok8
- http://gitready.com/

Σας ευχαριστώ πολύ

Ερωτήσεις;