ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Version control with Git & Github

Τριγάζη Ελισάβετ-Αθανασία
ΑΕΜ: 574
Συστήματα Παράλληλης & Κατανεμημένης Επεξεργασίας
Επιβλέπων καθηγητής: Δρ. Δασυγένης Μηνάς
https://arch.icte.uowm.gr/

Κοζάνη, Μάιος 2014

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

git

Version Control Systems

Ένα σύστημα ελέγχου εκδόσεων (version control system VCS – ή revision control system) είναι λογισμικό που επιτρέπει τη διαχείριση και παρακολούθηση αλλαγών που συμβαίνουν σε οποιοδήποτε έγγραφο, πρόγραμμα, web site κ.α.

Οι τρεις "γενιές" Συστημάτων Ελέγχου Έκδοσης

- Απλή, όταν μόνο ένα άτομο μπορεί να έχει πρόσβαση στα αρχεία σε κάθε δεδομένη στιγμή
  - RCS
- Σε δίκτυο αλλά ακόμα κεντρικοποιημένα
  - CVS
  - Subversion (SVN)

- Σε δίκτυο και κατανεμημένα
  - Git
  - Mercurial
  - Bazaar

Οφέλη από τη χρήση των VCS

- Οργάνωση αρχείων που έχουν δημιουργηθεί σε διαφορετικές χρονικές στιγμές (χωρίς συμβάσεις για ονοματοδοσία φακέλων βάσει ημερομηνίας)
- Ανάκτηση προηγούμενων εκδόσεων και σύγκριση μεταξύ εκδόσεων
- Διατήρηση log για τις αλλαγές που έχουν γίνει (εναλλακτικά θα έπρεπε να έχω ένα αρχείο στο οποίο να καταγράφω στοιχεία για κάθε αλλαγή που γίνεται)
- Δυνατότητα εργασίας πολλών ατόμων επί του ιδίου έργου, ανεξαρτήτως της γεωγραφικής τους θέσης ή χρονικής στιγμής
- Διαχείριση συγκρούσεων (ο χρήστης παρεμβαίνει για να τις επιλύσει)

Why Git

Το Git είναι ένα κατανεμημένο σύστημα ελέγχου εκδόσεων (distributed version control system – DVCS) που σημαίνει ότι:

- επιτρέπει την παράλληλη εργασία πολλαπλών ατόμων σε ένα έργο ακόμα και χωρίς την ύπαρξη σύνδεσης σε κεντρικό δίκτυο. Η εργασία τους μπορεί να καταχωρηθεί (push) στο έργο όταν είναι έτοιμη.
- είναι δωρεάν, σύμφωνα με την άδεια GPLv2
- είναι γρήγορο και υποστηρίζει πολύ μεγάλα projects π.χ. Linux Kernel
- είναι ασφαλές, η ιστορία και ο κώδικας είναι κρυπτογραφικά επικυρωμένα
- είναι modular και χρησιμοποιεί τα υπάρχοντα πρωτόκολλα (π.χ. HTTP, ssh, rsync)
- επιτρέπει τον έλεγχο χαμηλού επιπέδου (πολλοί τρόποι για να σπάσει το repository σας)

Implementation details

Τα δεδομένα οργανώνονται σε μια δομή δέντρου (παρόμοια με ένα σύστημα αρχείων). Οι δύο βασικές δομές:

- Δείκτη ή Cache
  - Η τρέχουσα κατάσταση ενός καταλόγου εργασίας.
  - Περιέχει όλες τις αλλαγές που πρέπει να γίνουν commit.
- Object database
  - Περιέχει όλη την ιστορία και τα αρχεία.
  - Περιλαμβάνει blobs, trees, commits και tags.
  - Κάθε αντικείμενο έχει ένα hash SHA1.

Οι ορισμοί των παραπάνω υπάρχουν στην επόμενη διαφάνεια.

Terms

Repository: Η δομή δεδομένων που αποθηκεύονται οι πληροφορίες. Το repository αποθηκεύεται τοπικά στο ίδιο directory με το project, σε ένα υποφάκελο με το όνομα .git

Commit: Αντιπροσωπεύει μία έκδοση του κώδικά μας, δηλαδή ένα σύνολο αρχείων για μία συγκεκριμένη χρονική στιγμή.
Head: Είναι ένας pointer στο τρέχον branch. Αυτό σημαίνει ότι θα είναι ο πατέρας αν γίνει ένα commit.

Working tree (or working directory): Το directory αυτό είναι ο φάκελος με όλα τα αρχεία σας μέσα. Ο χώρος στο local σύστημά σας που μπορείτε εύκολα να κάνετε modify κάποια αρχεία.

Index: Όταν δουλεύετε με αρχεία και έχετε κάνει κάποιες αλλαγές κάποια στιγμή θα γίνει ένα commit έτσι ώστε να ενημερωθεί η "βάση μας". Το Index είναι ένα στιγμιότυπο από το επόμενο commit που πρόκειται να γίνει. Σημειώστε πως όταν εκτελείτε git commit το commit κοιτά μόνο στο Index για τυχόν αλλαγές. Αν έχετε δημιουργήσει νέα αρχεία θα πρέπει να τα προσθέσετε.

Blobs: Ένας τύπος blob αποτελείται από πολλά bytes που μπορεί να είναι οτιδήποτε (π.χ. αρχείο κειμένου, πηγαίος κώδικας, εκτελέσιμο δυαδικό αρχείο ή εικόνα).

Repository location

- Στον προσωπικό μας Η/Υ:
  - είναι ο καλύτερος τρόπος για να δουλεύουμε κάποιο feature μόνοι μας ακόμα και αν θα το μοιραστούμε αργότερα,
  - όμως είναι δύσκολη η πρόσβαση στο repository από άλλους αλλά ακόμα και για μας από άλλη τοποθεσία
- Σε κάποιο server κοινό με τους συναδέλφους μας:
  - σε περίπτωση που όλα τα μέλη της ομάδας έχουν ήδη πρόσβαση στο server (με δικούς τους λογαριασμούς), μπορούμε εύκολα να διαχειριστούμε και τη πρόσβαση στο repo με τα υπάρχοντα δικαιώματα
  - συνεπώς, υπάρχει εμπιστοσύνη μεταξύ των μελών της ομάδας γιατί όλοι έχουν δικαίωμα εγγραφής
  - η πρόσβαση γίνεται με το local πρωτόκολο (`git clone /opt/git/project.git` ή `git clone file:///opt/git/project.git`)
- Στο Github:
  - η διαχείριση είναι ήδη ρυθμισμένη από το Github για όλους με λεπτομέρεια
  - ο κώδικας μας είναι αποθηκευμένος σε μια τρίτη υπηρεσία

Installing Git

- Εγκατάσταση σε Linux:

  Για διανομές βασισμένες σε Debian (όπως το ubuntu):

  ```
  sudo apt-get install git-core
  ```

  ενώ, για διανομές βασισμένες σε Fedora:

  ```
  yum install git-core
  ```

- Εγκατάσταση σε Windows XP/Vista/7 :

  Κατεβάζοντας απο τον σύνδεσμο http://code.google.com/p/msysgit το εκτελέσιμο αρχείο. Μόλις γίνει η εγκατάσταση θα πρέπει να έχετε ένα πρόγραμμα Git Bash, σαν ένα απλό κέλυφος όπου μπορείτε να εκτελείτε εντολές σχετικά με το git. Περισσότερες πληροφορίες στο: http://help.github.com/win-set-up-git/

- Εγκατάσταση σε Mac OS:

Χρησιμοποιείτε το GUI installer που το κατεβάζετε μέσω του συνδέσμου http://code.google.com/p/git-osx-installer. Αν έχετε ήδη εγκατεστημένα τα MacPosts μπορείτε να το κατεβάσετε με:

```
$ sudo port install git-core +svn +doc +bash_completion +gitweb
```

ή αν έχετε το homebrew

```
$ brew install git
```

First time setup

- Δίνετε ένα όνομα και ένα email όπως φαίνετε και παρακάτω:

```
git config --global user.name "type your name here"
git config --global user.email "Type your email here"
```

- Ρυθμίζετε το προεπιλεγμένο πρόγραμμα επεξεργασίας κειμένου:

```
git config --global core.editor <put your editor>
```

π.χ. nano, vim, kate,gedit

Άλλες ρυθμίσεις:

- Χρωματισμένη έξοδος εντολών git:

```
git config --global color.ui auto
```

- Κάθε στιγμή μπορείτε να ενημερώνεστε για τις ρυθμίσες που έχετε δώσει:

```
git config --list
```

Starting a git repository

- `cd` στον κατάλογο εργασίας που θα χρησιμοποιήσετε.
- Αρχικοποίηση του τρέχων φακέλου ώς αποθετήριο

`git init` (Δημιουργία του δικού μας repo)

`git clone <repo> [<path>]` (Αντιγραφή ενός project απο το github στον local δίσκο μας)

- Άν θέλετε να δημιουργήσετε ένα αποθετήριο χωρίς να είναι κατάλογος εργασίας:

```
git init --bare
```

Git workflow

Η βασική ροή εργασίας του Git είναι:

- Επεξεργασία των αρχείων
- Έλεγχος της κατάσταση του αποθετηρίου σας:

```
git-status
```

(εμφανίζει την κατάσταση του Working tree δηλαδή, μονοπάτια που έχουν διαφορά στο index και στο τρέχον HEAD commit. Τα πρώτα αρχεία μπορείτε να τα κάνετε commit. Τα τρίτα και τέταρτα μπορείτε να τα κάνετε commit αφού γίνει πρώτα add.)

- Πρόσθεση του περιεχομένου του αρχείου στο index και προετοιμασία για το επόμενο commit:

```
git-add
```

- Κάνοντας commit τις αλλαγές:

```
git commit
```

- Προαιρετικά: Μεταφορτώστε τις αλλαγές σας:

```
git push origin master
```

.

Other commands

Εντολές που επιλύουν κάποιες φορές περίπλοκα θέματα

- Αν έχετε προσθέσει κάποιον συνεργάτη (collaborator) στο GitHub για το αποθετήριο και έχει προβεί σε κάποιες αλλαγές, ενημερώνετε το αποθετήριο στον υπολογιστή μας με την εντολή:

```
git pull --rebase
```

- Αν για οποιοδήποτε λόγο θέλουμε να επαναφέρουμε το αποθετήριο στην κατάσταση του τελευταίου commit:

```
git reset
```

- Επαναφορά ορισμένων υπάρχοντων commits:

```
git revert
```

```
--continue,  --quit, --abort
```

- Κρύβει τις αλλαγές που έχετε κάνει ώστε να είστε σε ένα “καθαρό” κατάλογο:

```
git stash
```

Git status

Εμφανίζει την τρέχουσα κατάσταση του repository

- αλλαγές στο index (θα γίνουν commit)
- αλλαγές στο index (δεν θα γίνουν commit)
- αρχεία που δεν έχουν προστεθεί στο index
- αρχεία που περιέχονται στο .gitignore δεν θα εμφανίζονται όταν κάνουμε git status

Στο ακόλουθο screenshot

έχουν γίνει αλλαγές και

στα τέσσερα αρχεία:

```
eliza@eliza-PC ~/test-repo $ ls
file1.txt  file2.txt  ignored-file1.txt  README
eliza@eliza-PC ~/test-repo $ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   README
#
# 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)
#
#       modified:   file1.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       file2.txt
eliza@eliza-PC ~/test-repo $ cat .gitignore
ignored-file1.txt
eliza@eliza-PC ~/test-repo $
```

Exploring the repository (1)

`git log`

- Εμφανίζει πληροφορίες για τα commit που έχουν γίνει.
- Τα πρώτα 7 γράμματα του αθροίσματος SHA1 μπορούν να προσδιορίσουν το commit.

`git log --grep=<pattern> ή git log -S'<pattern>`

Θα φιλτράρει τα commits από το συγκεκριμένο patern.

```
eliza@eliza-PC ~/test-repo $ git log
commit faa02b8a4348a94723d50ac3d5b3391213edf716
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Tue Apr 8 19:54:09 2014 +0300

    Add .gitignore file

    Body.

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>

commit 18ebc167a1a75e5cd480bd5ea40d57ac8c03b662
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Tue Apr 8 19:48:39 2014 +0300

    Second commit

    Body.

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>

commit 8cfe45ce94f285e0e1aaf0558796f22cd7c23d2c
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Tue Apr 8 19:46:36 2014 +0300

    Initial commit

    Body of commit.

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>
```

Exploring the repository (2)

`git diff`

Εμφανίζει διαφορές στα commit που έχουν γίνει και στο working tree με τα index.

- `git diff [<path>]`
- Οι αλλαγές σε σχέση με το INDEX.
- `git diff --cached [<path>]`
- Οι αλλαγές στο INDEX (to be commited).
- `git diff <commit> [<commit>] [<path>]`
  Αλλαγές μεταξύ δύο commit.
  Εάν παραλείπεται, το δεύτερο εκτελεί την προεπιλογή HEAD.
- `git show <commit>`
  Θα εμφανίσει όλες τις αλλαγές που επήλθαν στο δεδομένο commit.

```
eliza@eliza-PC ~/test-repo $ git diff
diff --git a/README b/README
index 876acd2..236bf03 100644
--- a/README
+++ b/README
@@ -1 +1,2 @@
 Dummy README file.
+A change.
diff --git a/file1.txt b/file1.txt
index 50155c6..0d85444 100644
--- a/file1.txt
+++ b/file1.txt
@@ -1 +1,2 @@
-text text
+text text text
+A change.
```

Exploring branches

- `git branch`

Εμφανίζει σε λίστα τα branches που υπάρχουν.

- `git branch <branch_name>`

Δημιουργεί νέο branch αλλά δεν αλλάζει σε αυτό.

- `git checkout <branch_name>`

Εναλλαγή μεταξύ branches.

- `git branch -d [head]`

Διαγραφή κάποιου branch όταν πάψει να μας χρειάζεται.

Merging

Αφού αναπτύξουμε τα νέα features στα ξεχωριστά branches πρέπει κάπως να τα εισάγουμε στο master branch: `git merge [head]`

Τι κάνει το merge;

- Εντοπίζει το κοινό πρόγονο (ancestor) του HEAD και του head που θα γίνει merge.
- Αν ο πρόγονος == merge τότε δε κάνει τίποτα
  Αν ο πρόγονος == HEAD τότε fast forward merge
- Διαφορετικά εντοπίζει διαφορές μεταξύ πρόγονου και merge.
- Δοκιμάζει να τα συγχωνεύσει σε ένα αρχείο.
- Αν δεν υπάρχει σύγκρουση δημιουργεί νέο commit.
- Αν υπάρχει σύγκρουση δείχνει που υπάρχει πρόβλημα και ενημερώνει το χρήστη.

Merging Branches

- `git merge <commit>`

Ενώνει δύο η περισσότερα branches μαζί. Για παράδειγμα το master με κάποιο δεύτερο που δημιουργήσατε.

- `git merge --abort`

Αναιρεί μια συγχώνευση σε εξέλιξη (πάντα κάνετε commit πριν από τη συγχώνευση).

- `git cherry-pick <commit>`

Επιλέξτε μόνο μερικά commits για τη συγχώνευση.

Solving Conflicts

Όταν τα branches έχουν αλλάξει το ίδιο κομμάτι του κώδικα, προκύπτουν συγκρούσεις οι οποίες επιλύονται:

- Χειροκίνητα ή χρησιμοποιώντας το εργαλείο mergetool: `git mergetool`
- Προσθέτωντας τα αρχεία στο INDEX.
- Κάνοντας commit στα αρχεία τα οποία έχετε επιλύσει τις συγκρούσεις.

Sharing and update projects

- `git fetch <repo> [<ref>]` e.g. `git fetch origin master`
  Φέρνουμε-κατεβάζουμε από το remote repository origin το branch master.
- `git pull` e.g. `git pull origin master`

  Φέρνουμε-κατεβάζουμε από το remote repository origin το branch master, και στη συνέχεια κάνουμε και merge.

  e.g. `git pull origin master --rebase`

Με το rebase ζητάμε από το git να τροποποιήσει το τοπικό μας αποθετήριο και να κάνει αυτά τα commit που δεν υπάρχουν ακόμα στο δημόσιο αποθετήριο, να εμφανιστούν με τέτοιο τρόπο ώστε να περάσουν πρώτα στο ιστορικό (του τοπικού αποθετηρίου) τα commit που υπήρχαν στο δημόσιο αποθετήριο, και τα commit του τοπικού να εμφανιστούν πιο πρόσφατα στο ιστορικό.

- `git push [<repo>] :[<ref>]` e.g. `git push -u origin master`

Ανανεώνουμε κάποιο public αποθετήριο.

`git-remote` e.g. `git remote add origin https://github.com/elitrigazi/testrepo.git`

Διαχείριση αποθετηρίων στα οποία κάποιο απο τα branches είναι σε trackignmode.

Git reset

Αυτή η εντολή επαναφέρει το τρέχον HEAD στην καθορισμένη κατάσταση.

- `git reset <commit> <file>` : Opposite of `git add`

  `git reset HEAD <file>`

- `git reset <commit>`

`--soft` Αλλάζει το HEAD, αφήνει το INDEX και τα αρχεία άθικτα.

`--mixed` Αλλάζει το HEAD και το INDEX, αφήνει τα αρχεία άθικτα.

`--hard` Επαναφορά του αποθετηρίου στην κατάσταση του commit που πήρε σαν παράμετρο (χρησιμοποιείστε το με πολύ προσοχή!)

Παράδειγμα workflow

Στο παρακάτω παράδειγμα:

- αρχικοποιούμε ένα αποθετήριο (git init)
- προσθέτουμε το hello.c (git add hello.c, git commit)
- προσθέτουμε ένα Makefile που κάνει μόνο compile το hello.c (git add Makefile, git commit)
- τροποποιούμε το Makefile προσθέτοντας τη λειτουργία clean (git add Makefile, git commit)
- τροποποιούμε το hello.c ώστε να τυπώνει δύο φορές το “Hello” (git add hello.c, git commit)
- κάνουμε revert το τελευταίο commit για να ακυρώσουμε τη τελευταία τροποποίηση
  - για να κάνουμε revert χρειαζόμαστε τουλάχιστον τα πρώτα επτά ψηφία του SHA1 του commit που θέλουμε να κάνουμε revert
  - για να τα βρούμε, χρησιμοποιούμε την εντολή git log (με την εντολή git show μπορούμε να κάνουμε επισκόπηση ενός commit)

Πριν από κάθε εντολή add, commit, pull, merge, revert κλπ συνίσταται να χρησιμοποιούμε τις εντολές status, show και diff για να είμαστε σίγουροι τι πρόκειται να κάνουμε.

Παράδειγμα workflow (init-add-commit)

```
[eliza@eliza-PC examples]$ mkdir hellorepo
[eliza@eliza-PC examples]$ cd hellorepo
[eliza@eliza-PC hellorepo]$ git init
Initialized empty Git repository in /home/eliza/sxolh/parsys/git/examples/hellorepo/.git/
[eliza@eliza-PC hellorepo]$
```

```
[eliza@eliza-PC hellorepo]$ nano hello.c
[eliza@eliza-PC hellorepo]$ gcc -o hello hello.c
[eliza@eliza-PC hellorepo]$ chmod +x hello
[eliza@eliza-PC hellorepo]$ ./hello
Hello
[eliza@eliza-PC hellorepo]$
```

```
[eliza@eliza-PC hellorepo]$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       hello
#       hello.c
nothing added to commit but untracked files present (use "git add" to track)
[eliza@eliza-PC hellorepo]$ git add hello.c
[eliza@eliza-PC hellorepo]$ git commit -s
```

```
Initial commit: Add greeter file

Add a greeter file which prints "Hello".

Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   hello.c
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       hello
```

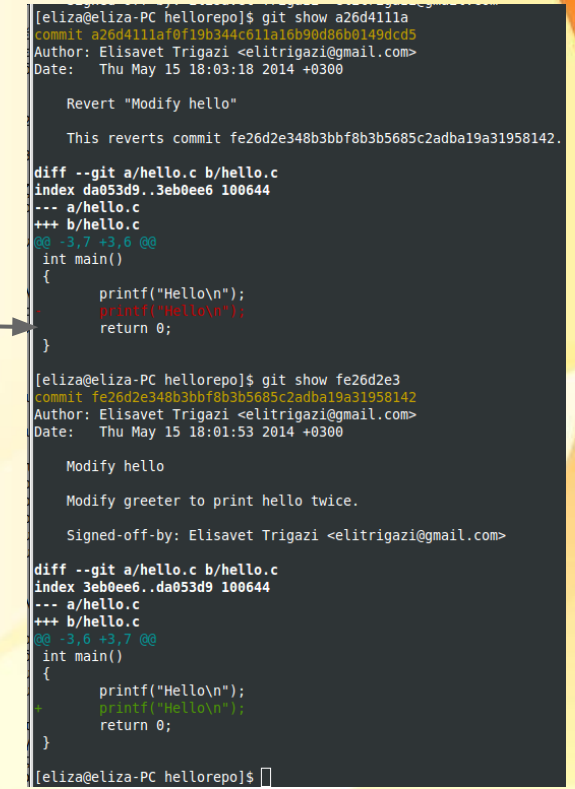
Παράδειγμα workflow (add-commit-modify-commit)

```
[eliza@eliza-PC hellorepo]$ git add Makefile
[eliza@eliza-PC hellorepo]$ git commit -s
[master 1dfefc5] Add Makefile
 1 file changed, 2 insertions(+)
 create mode 100644 Makefile
[eliza@eliza-PC hellorepo]$ git log
commit 1dfefc5ce1d6185233f70de3793ada2995712f9a
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Thu May 15 17:04:27 2014 +0300

    Add Makefile

    Add a Makefile to compile hello.c.

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>

commit 75fb4a5004e1147670a0562d4707b2e093b3fe4d
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Thu May 15 16:58:37 2014 +0300

    Initial commit: Add greeter file

    Add a greeter file which prints "Hello".

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>
[eliza@eliza-PC hellorepo]$
```

```
[eliza@eliza-PC hellorepo]$ git log
commit 91f8ae5763e8bcb9df27f8a3a3fb347e6c87e2dd
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Thu May 15 17:06:51 2014 +0300

    Modify Makefile to clean binaries

    Modify Makefile to remove the hello binary.

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>

commit 1dfefc5ce1d6185233f70de3793ada2995712f9a
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Thu May 15 17:04:27 2014 +0300

    Add Makefile

    Add a Makefile to compile hello.c.

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>

commit 75fb4a5004e1147670a0562d4707b2e093b3fe4d
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Thu May 15 16:58:37 2014 +0300

    Initial commit: Add greeter file

    Add a greeter file which prints "Hello".

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>
[eliza@eliza-PC hellorepo]$ git show 91f8ae5
commit 91f8ae5763e8bcb9df27f8a3a3fb347e6c87e2dd
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Thu May 15 17:06:51 2014 +0300

    Modify Makefile to clean binaries

    Modify Makefile to remove the hello binary.

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>

diff --git a/Makefile b/Makefile
index 5728321..1c37a60 100644
--- a/Makefile
+++ b/Makefile
@@ -1,2 +1,4 @@
 all:
         gcc -o hello hello.c
+clean:
+       rm -rf hello
[eliza@eliza-PC hellorepo]$
```

Παράδειγμα workflow (revert)

```
[eliza@eliza-PC hellorepo]$ git revert fe26d2e
[reverttest a26d411] Revert "Modify hello"
 1 file changed, 1 deletion(-)
[eliza@eliza-PC hellorepo]$ git log
commit a26d4111af0f19b344c611a16b90d86b0149dcd5
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Thu May 15 18:03:18 2014 +0300

    Revert "Modify hello"

    This reverts commit fe26d2e348b3bbf8b3b5685c2adba19a31958142.

commit fe26d2e348b3bbf8b3b5685c2adba19a31958142
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Thu May 15 18:01:53 2014 +0300

    Modify hello

    Modify greeter to print hello twice.

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>
```



```
[eliza@eliza-PC hellorepo]$ git show a26d4111a
commit a26d4111af0f19b344c611a16b90d86b0149dcd5
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Thu May 15 18:03:18 2014 +0300

    Revert "Modify hello"

    This reverts commit fe26d2e348b3bbf8b3b5685c2adba19a31958142.

diff --git a/hello.c b/hello.c
index da053d9..3eb0ee6 100644
--- a/hello.c
+++ b/hello.c
@@ -3,7 +3,6 @@
 int main()
 {
        printf("Hello\n");
-       printf("Hello\n");
        return 0;
 }

[eliza@eliza-PC hellorepo]$ git show fe26d2e3
commit fe26d2e348b3bbf8b3b5685c2adba19a31958142
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Thu May 15 18:01:53 2014 +0300

    Modify hello

    Modify greeter to print hello twice.

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>

diff --git a/hello.c b/hello.c
index 3eb0ee6..da053d9 100644
--- a/hello.c
+++ b/hello.c
@@ -3,6 +3,7 @@
 int main()
 {
        printf("Hello\n");
+       printf("Hello\n");
        return 0;
 }

[eliza@eliza-PC hellorepo]$
```

Git στο pleiades

- Στο pleiades μπορείτε να δημιουργήσετε ένα git repository και δίνοντας τη διαδρομή του σε κάποιο συμφοιτητή σας μπορεί να κατεβάσει το κώδικα.
- έστω ότι το repo σας είναι στο `$HOME/testrepo` δηλαδή

```
/zstorage/home/ictest00XXX/testrepo
```

- μπορεί κάποιος να το κατεβάσει δίνοντας:

```
git clone /zstorage/home/ictest00574/testrepo
```

  - Σε αυτό το repo δεν έχετε δικαίωμα εγγραφής διότι βρίσκεται στο HOME κάποιου άλλου χρήστη.
- Για να δουλέψετε συνεργατικά, πρέπει να δημιουργηθεί κάποιος κοινόχρηστος φάκελος.

Ολοκληρωμένο σενάριο χρήσης στο pleiades

Για να έχουμε πρόσβαση στο pleiades από το σπίτι (ή εκτός του πανεπιστημίου), πρέπει να είμαστε συνδεδεμένοι με vpn. Στο παρακάτω παράδειγμα συνεργάζονται δύο φοιτητές έχοντας ένα αποθετήριο στο pleiades για να ανταλλάσσουν τον κώδικα τους. Έστω ότι ο πρώτος φοιτητής έχει ένα αποθετήριο στον προσωπικό του Η/Υ και θέλει να τον μοιραστεί:

```
[eliza@eliza-PC examples]$ mkdir testrepo
[eliza@eliza-PC examples]$ ls
testrepo
[eliza@eliza-PC examples]$ cd testrepo
[eliza@eliza-PC testrepo]$ vi hello.py
[eliza@eliza-PC testrepo]$ python hello.py
Hello
[eliza@eliza-PC testrepo]$ git init
Initialized empty Git repository in /home/eliza/sxolh/parsys/git/examples/testrepo/.git/
[eliza@eliza-PC testrepo]$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       hello.py
nothing added to commit but untracked files present (use "git add" to track)
[eliza@eliza-PC testrepo]$ git add hello.py
[eliza@eliza-PC testrepo]$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   hello.py
#
[eliza@eliza-PC testrepo]$ git commit -s
```

```
[eliza@eliza-PC testrepo]$ git commit -s
[master (root-commit) 2a1053a] Initial commit: Add a greeter file
 1 file changed, 1 insertion(+)
 create mode 100644 hello.py
[eliza@eliza-PC testrepo]$ git log
commit 2a1053a90fe9bb4446be910e26a312ed7be91592
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Thu May 15 15:03:05 2014 +0300

    Initial commit: Add a greeter file

    Add a hello file which prints "Hello".

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>
[eliza@eliza-PC testrepo]$
```

Δημιουργία αποθετηρίου στο Pleiades

```
ictest00574@pleiades:/mnt/nas/574/parsys/git-examples$ mkdir testrepo
ictest00574@pleiades:/mnt/nas/574/parsys/git-examples$ cd testrepo
ictest00574@pleiades:/mnt/nas/574/parsys/git-examples/testrepo$ git init --bare --shared
Initialized empty shared Git repository in /mnt/nas/574/parsys/git-examples/testrepo/
ictest00574@pleiades:/mnt/nas/574/parsys/git-examples/testrepo$
```

Αρχικοποιούμε ένα αποθετήριο στον κοινόχρηστο κατάλογο `/mnt/nas` δίνοντας τις παραμέτρους `--bare` `--shared`. Με το --bare δηλώνουμε ότι το αποθετήριο θα είναι κενό και με το --shared ότι οι χρήστες που ανήκουν στο ίδιο group μπορούν να δημοσιεύουν σε αυτό το αποθετήριο.

Από το man page:

`--bare`

Create a bare repository. If GIT_DIR environment is not set, it is set to the current working directory.

`--shared`

[=(false|true|umask|group|all|world|everybody|0xxx)]

Specify that the git repository is to be shared amongst several users. This allows users belonging to the same group to push into that repository. When specified, the config variable “core.sharedRepository" is set so that files and directories under $GIT_DIR are created with the requested permissions. When not specified, git will use permissions reported by umask(2).

Προσθήκη του απομακρυσμένου αποθετηρίου στο τοπικό και push του κώδικα

```
[eliza@eliza-PC testrepo]$ git remote add pleiades-testrepo ssh://ictest00574@pleiades.icte.uowm.gr/mnt/nas/574/parsys/git-examples/testrepo
[eliza@eliza-PC testrepo]$ git remote
pleiades-testrepo
[eliza@eliza-PC testrepo]$
```

```
[eliza@eliza-PC testrepo]$ git branch
* master
```

```
[eliza@eliza-PC testrepo]$ git push pleiades-testrepo master
Counting objects: 3, done.
Writing objects: 100% (3/3), 286 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To ssh://ictest00574@pleiades.icte.uowm.gr/mnt/nas/574/parsys/git-examples/testrepo
 * [new branch]      master -> master
[eliza@eliza-PC testrepo]$
```

Pull ή clone από τον δεύτερο φοιτητή

Σε περίπτωση που θέλουμε αυτούσιο το αποθετήριο κάνουμε clone:

(στον home μας στο Pleiades)

```
ictest00570@pleiades:~$ git clone file:///mnt/nas/574/parsys/git-examples/testrepo/
Cloning into 'testrepo'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (3/3), done.
ictest00570@pleiades:~$
```

(στο Η/Υ μας μέσω ssh)

```
test1@eliza-PC ~ $ git clone ssh://ictest00570@pleiades.icte.uowm.gr/mnt/nas/574/parsys/git-examples/testrepo/
Cloning into 'testrepo'...
ictest00570@pleiades.icte.uowm.gr's password:
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (3/3), done.
test1@eliza-PC ~ $
```

Αν δουλεύαμε ήδη σε κάποιο αποθετήριο θα μπορούσαμε να προσθέσουμε το απομακρυσμένο ως εξής: `git remote add pleiades-testrepo file:///mnt/nas/574/parsys/git-examples/testrepo` ή

```
git remote add pleiades-testrepo \
> ssh://ictest00570@pleiades.icte.uowm.gr/mnt/nas/574/parsys/git-examples/testrepo
```

και στη συνέχεια `git pull pleiades-testrepo master` (ή σε όποιο άλλο branch θέλουμε)

Αλλαγές και push στο απομακρυσμένο αποθετήριο

```
ictest00570@pleiades:~/testrepo$ nano hello.py
ictest00570@pleiades:~/testrepo$ python hello.py
Hello, World!
ictest00570@pleiades:~/testrepo$ 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)
#
#       modified:   hello.py
#
no changes added to commit (use "git add" and/or "git commit -a")
ictest00570@pleiades:~/testrepo$ git add hello.py
ictest00570@pleiades:~/testrepo$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   hello.py
#
ictest00570@pleiades:~/testrepo$ git commit -s
```

```
ictest00570@pleiades:~/testrepo$ git commit -s
[master 3a4f47f] Modify hello file
 1 file changed, 1 insertion(+), 1 deletion(-)
ictest00570@pleiades:~/testrepo$ git log
commit 3a4f47fc788b46a7f89960cd53609f9c0a755922
Author: st570 <st570@icte.uowm.gr>
Date:   Thu May 15 15:46:28 2014 +0300

    Modify hello file

    Modify hello file to greet the World.

    Signed-off-by: st570 <st570@icte.uowm.gr>

commit 2a1053a90fe9bb4446be910e26a312ed7be91592
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Thu May 15 15:03:05 2014 +0300

    Initial commit: Add a greeter file

    Add a hello file which prints "Hello".

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>
ictest00570@pleiades:~/testrepo$
```

```
ictest00570@pleiades:~/testrepo$ git remote
origin
ictest00570@pleiades:~/testrepo$ git branch
* master
ictest00570@pleiades:~/testrepo$ git push origin master
Counting objects: 5, done.
Writing objects: 100% (3/3), 301 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To file:///mnt/nas/574/parsys/git-examples/testrepo/
   2a1053a..3a4f47f  master -> master
ictest00570@pleiades:~/testrepo$
```

Pull από τον πρώτο φοιτητή

```
[eliza@eliza-PC testrepo]$ git pull pleiades-testrepo master
From ssh://pleiades.icte.uowm.gr/mnt/nas/574/parsys/git-examples/testrepo
 * branch            master     -> FETCH_HEAD
Updating 2a1053a..3a4f47f
Fast-forward
 hello.py | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
[eliza@eliza-PC testrepo]$ git log
commit 3a4f47fc788b46a7f89960cd53609f9c0a755922
Author: st570 <st570@icte.uowm.gr>
Date:   Thu May 15 15:46:28 2014 +0300

    Modify hello file

    Modify hello file to greet the World.

    Signed-off-by: st570 <st570@icte.uowm.gr>

commit 2a1053a90fe9bb4446be910e26a312ed7be91592
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Thu May 15 15:03:05 2014 +0300

    Initial commit: Add a greeter file

    Add a hello file which prints "Hello".

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>
[eliza@eliza-PC testrepo]$ python hello.py
Hello, World!
[eliza@eliza-PC testrepo]$
```

Όταν κάναμε pull στην ουσία το git έκανε fetch και merge. Το merge που έγινε ήταν fast-forward, το happy scenario του merge. Σε διαφορετική περίπτωση αν είχαμε κάνει αλλαγές μέχρι ο δεύτερος φοιτητής να κάνει push και οι αλλαγές επηρέαζαν τις ίδιες γραμμές θα είχαμε συγκρούσεις.

Όταν δεν έχουμε διακομιστή git μπορούμε να χρησιμοποιήσουμε το github!

Εισαγωγή

Το github είναι ένα social repository για projects ανοιχτού κώδικα που βασίζονται πάνω στο Git.

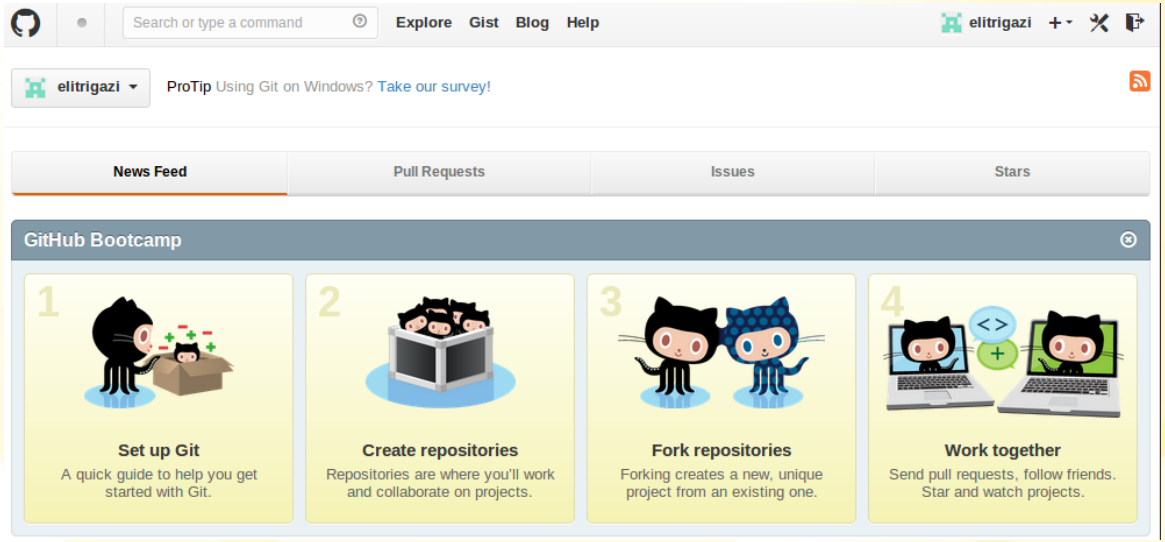
- Κύρια εφαρμογή του είναι να κάνει εύκολη την διαμοίραση κώδικα και την συνεργασία σε projects.
- Χρησιμοποιείται από 3,5 εκατομμύρια και πάνω χρήστες με περισσότερα από 10 εκατομμύρια αποθετήρια.
- Προσφέρει ένα βολικό τρόπο για να κρατήσει όλα τα αποθετήρια σας στο cloud.
  - Μπορείτε να έχετε πρόσβαση στα αποθετήρια σας όπου και αν βρίσκεστε.
  - Μπορείτε να μοιράζεστε τα project σας με οποιονδήποτε μόνο με την αποστολή ενός link.

Μερικά χαρακτηριστικά:

- διεπαφή του χρήστη με τα γραφικά
- απεριόριστα δημόσια αποθετήρια
- 5 δωρεάν ιδιωτικά αποθετήρια για τους φοιτητές
  https://education.github.com/

Εγκατάσταση

Δημιουργήστε τον προσωπικό σας λογαριασμό:



Περισσότερες πληροφορίες:

https://help.github.com/articles/set-up-git

Forking a repository

The Pull model

- Στο προφίλ του project που θέλουμε να κάνουμε προσθήκες επιλέγουμε το κουμπί fork.
- Αυτό δημιουργεί ένα αντίγραφο του project στο προφίλ μας.
- Κατεβάζουμε το project από το προφίλ μας με `git clone` και κάνουμε τις αλλαγές που θέλουμε προσέχοντας να μην πειράξουμε καθόλου κώδικα που δεν χρειάζεται (όπως στοίχιση και indentation).
- Κάνουμε commit τις αλλαγές προσθέτοντας κατατοπιστικό τίτλο και κείμενο στο commit message.
- Κάνουμε push τις αλλαγές στο project στο προγίλ μας στο github.
- Στέλνουμε ένα Pull request στον ιδιοκτήτη του αρχικού project για να δεχτεί τις αλλαγές μας.

Working with others

Συνήθως όταν δουλεύετε με άλλους στο Github:

- Κατεβάζετε το τρέχον HEAD από το κεντρικό αποθετήριο.
- Κάνετε τις αλλαγές σας.
- Κάνετε commit τις αλλαγές στον τοπικό σας repository.
- Βεβαιωθείτε ότι κάποιος άλλος στην ομάδα σας δεν έχει κάνει uptaded το κεντρικό αποθετήριο.
- Αφού το πήρατε μεταφορτώσετε τις αλλαγές στο κεντρικό αποθετήριο.

Εάν το κεντρικό αποθετήριο έχει αλλάξει από τότε που το πήρατε,

- είναι δική σας ευθύνη να συγχωνεύσετε δύο εκδόσεις σας
- συχνά το git μπορεί να κάνει αυτό για σας, αν δεν υπάρχουν ασύμβατες αλλαγές.

Typical workflow

- `git pull <remote_repository>`

  Ανακτήστε τις αλλαγές από έναν απομακρυσμένο χώρο αποθήκευσης και τη συγχώνευσή τους στο αποθετήριο σας πάντα χρήστη αυτό πριν αρχίσετε να κάνετε αλλαγές. Εκτελείτε πάντα αυτό πριν αρχίσετε να κάνετε αλλαγές.

- `git status`

- `git commit`

- `git push -u <remote_repository> <branch>`

Το online αποθετήριο σας είναι πλέον ενημερωμένο!

A pull request workflow

Στο παρακάτω παράδειγμα συνεργάζονται δύο φοιτητές για να ανταλλάξουν τον κώδικα τους κάνοντας pull request μέσω του github.

- Ο πρώτος φοιτητής έχει κώδικα στη διάθεση του και δημιουργεί ένα project στο github όπου και κάνει push τον κώδικα.
- Ο δεύτερος φοιτητής κάνει fork το project, κάνει αλλαγές, τις κάνει push στο forked project του και κάνει pull request στο πρώτο φοιτητή.
- Ο πρώτος φοιτητής δέχεται το αίτημα και κάνει merge τις αλλαγές.

Αυτό το workflow χρησιμοποιείται κυρίως όταν συνεισφέρουμε σε project τρίτων που δεν γνωρίζουμε. Στη περίπτωση που κάνουμε μια εργασία θα μπορούσαμε να είμαστε μέλη όλοι στο ίδιο project. Αν είμαστε μέλη όλοι στο ίδιο project και έχουμε write access επιταχύνουμε τη διαδικασία αλλά πρέπει να υπάρχει εμπιστοσύνη μεταξύ των μελών. Σε αυτή τη περίπτωση ο κύκλος εργασίας είναι ίδιος με το προηγούμενο παράδειγμα στο pleiades.

Initialize repo

```
[eliza@eliza-PC testrepo]$ git init
Initialized empty Git repository in /home/eliza/sxolh/parsys/git/examples/github
test/testrepo/.git/
[eliza@eliza-PC testrepo]$
```

```
[eliza@eliza-PC testrepo]$ nano hello.py
[eliza@eliza-PC testrepo]$ python hello.py
Hello
[eliza@eliza-PC testrepo]$
```

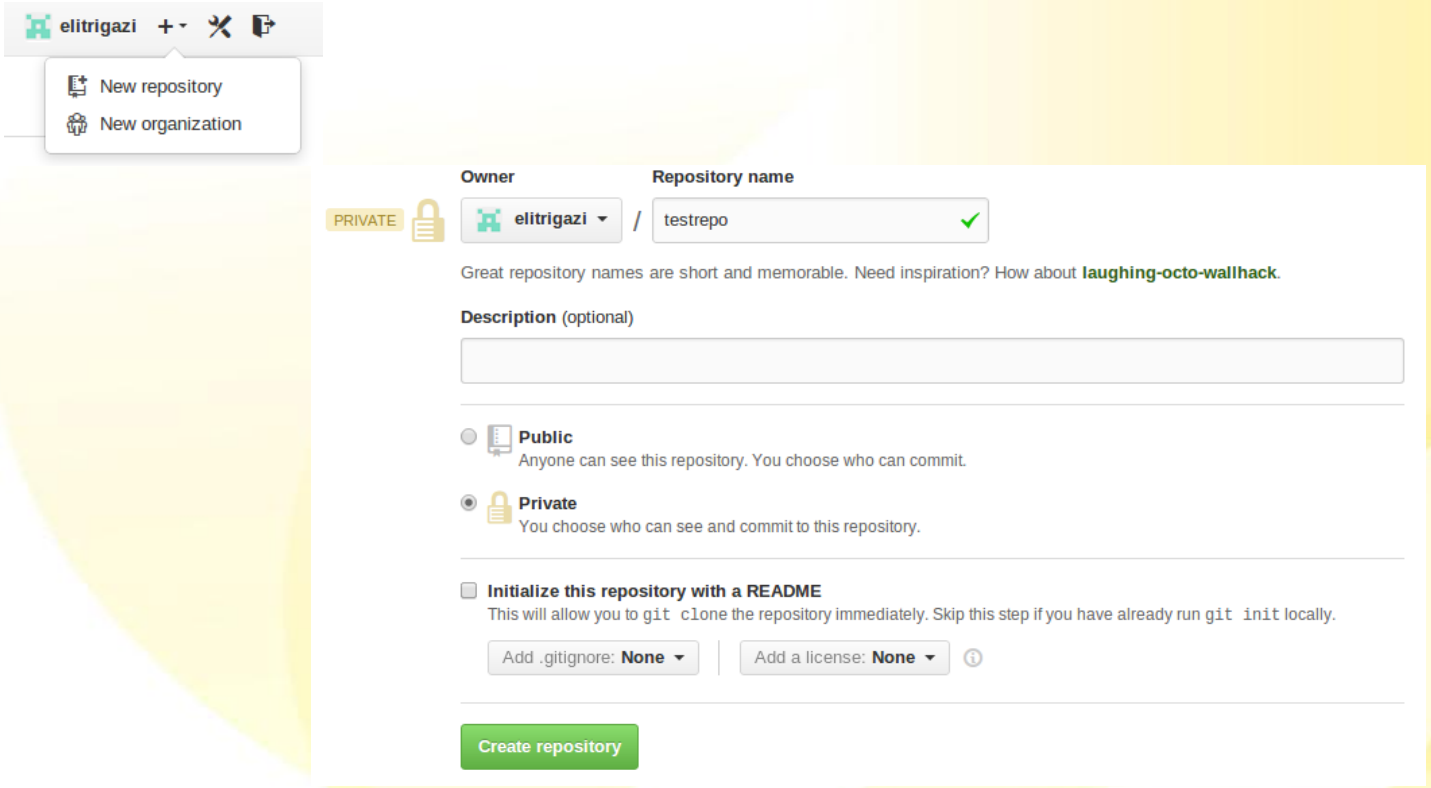
Πριν και μετά το git add πάντα κάνουμε git status, log, diff και branch (αν έχουμε πολλά branches) για να είμαστε σίγουροι τι κάνουμε!

```
[eliza@eliza-PC testrepo]$ git add hello.py
```

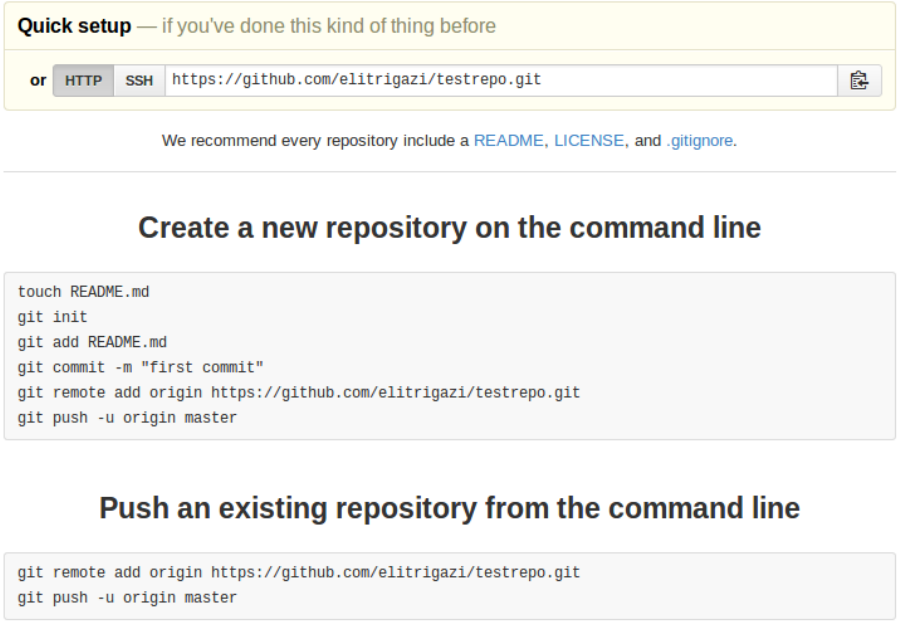
```
  GNU nano 2.2.6            File: .git/COMMIT_EDITMSG

Initial commit: Add greeter

Add hello file which prints "Hello".

Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   hello.py
#
```

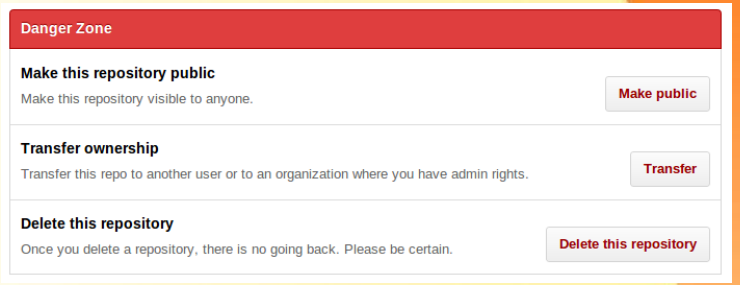
```
[eliza@eliza-PC testrepo]$ git commit -s
[master (root-commit) 44b9457] Initial commit: Add greeter
 1 file changed, 1 insertion(+)
 create mode 100644 hello.py
```

Δημιουργία project στο github



Project urls





Από τις ρυθμίσεις του project μπορούμε να το κάνουμε δημόσιο και αντίστροφα. Σε αυτό το παράδειγμα το project μας θα είναι δημόσιο.

Μόλις δημιουργήσουμε το project βλέπουμε τα url και τις εντολές που χρειαζόμαστε.

Push στο απομακρυσμένο (remote) αποθετήριο

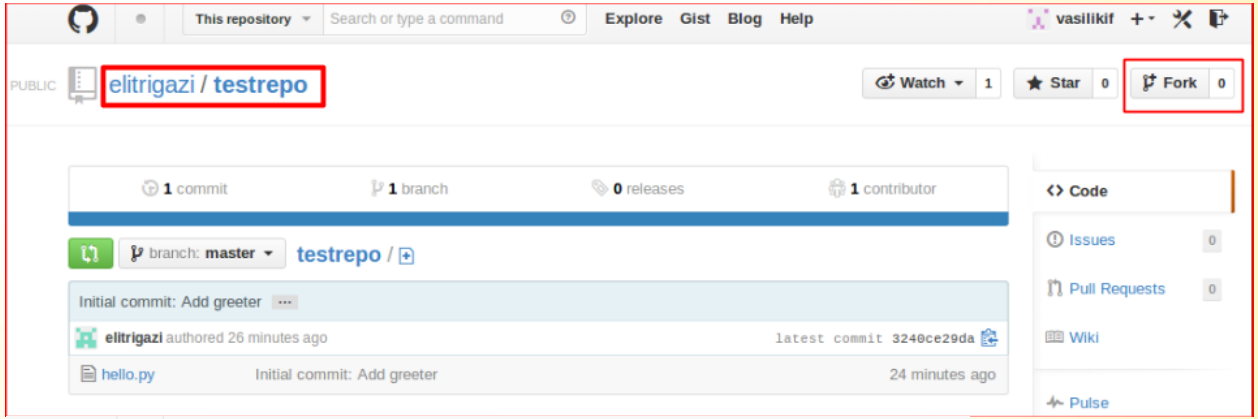
```
[eliza@eliza-PC testrepo]$ git remote add origin https://github.com/elitrigazi/t
estrepo.git
```

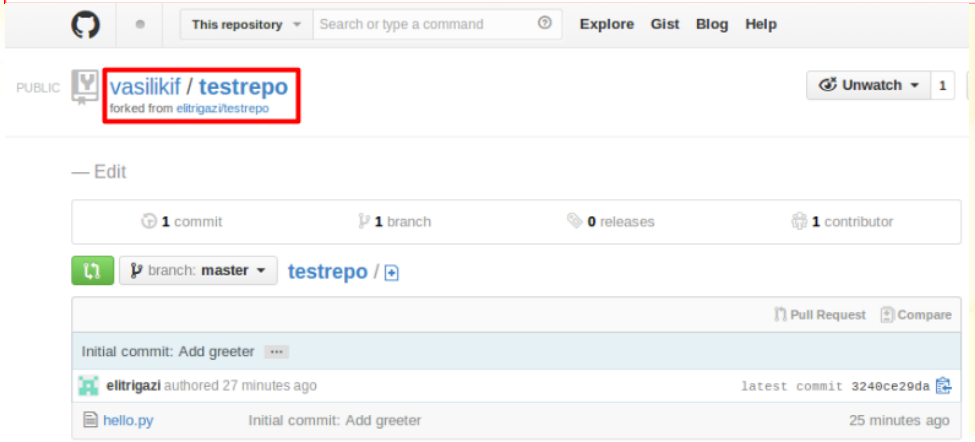
```
[eliza@eliza-PC testrepo]$ git remote show origin
Username for 'https://github.com': elitrigazi
Password for 'https://elitrigazi@github.com':
* remote origin
  Fetch URL: https://github.com/elitrigazi/testrepo.git
  Push  URL: https://github.com/elitrigazi/testrepo.git
  HEAD branch: (unknown)
```

```
[eliza@eliza-PC testrepo]$ git branch
* master
[eliza@eliza-PC testrepo]$ git push origin master
Username for 'https://github.com': elitrigazi
Password for 'https://elitrigazi@github.com':
Counting objects: 3, done.
Writing objects: 100% (3/3), 287 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/elitrigazi/testrepo.git
 * [new branch]      master -> master
```

Αν δεν έχουμε προσθέσει το δημόσιο κλειδί μας στο github θα χρειαστεί να δώσουμε το username και password.

Fork project





Ο δεύτερος φοιτητής κάνει fork το project του πρώτου δημιουργώντας ένα νέο αντίγραφο στον λογαριασμό του.

Clone -> edit -> add -> commit

```
[test1@eliza-PC examples]$ git clone https://github.com/vasilikif/testrepo.git
Cloning into 'testrepo'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 3 (delta 0)
Unpacking objects: 100% (3/3), done.
[test1@eliza-PC examples]$ cd testrepo
[test1@eliza-PC testrepo]$ git log
commit 3240ce29da17a252cfb721c45f8045f60f030fc9
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Mon May 19 15:39:16 2014 +0300

    Initial commit: Add greeter

    Add hello file which prints "Hello".

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>
[test1@eliza-PC testrepo]$
```

```
[test1@eliza-PC testrepo]$ nano hello.py
[test1@eliza-PC testrepo]$ python hello.py
Hello, github!
```

```
[test1@eliza-PC testrepo]$ 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)
#
#       modified:   hello.py
#
no changes added to commit (use "git add" and/or "git commit -a")
[test1@eliza-PC testrepo]$ git diff
diff --git a/hello.py b/hello.py
index 9df8753..9ab7f08 100644
--- a/hello.py
+++ b/hello.py
@@ -1 +1 @@
-print "Hello"
+print "Hello, github!"
```

```
[test1@eliza-PC testrepo]$ git add hello.py
[test1@eliza-PC testrepo]$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   hello.py
#
[test1@eliza-PC testrepo]$
```

```
Modify greeter

Modify hello file to greet github.

Signed-off-by: st570 <st0570@icte.uowm.gr>

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#   (use "git push" to publish your local commits)
#
# Changes to be committed:
#   (use "git reset HEAD^1 <file>..." to unstage)
#
#       modified:   hello.py
#
```

```
[test1@eliza-PC testrepo]$ git commit -s
[master 85d2a69] Modify greeter
 1 file changed, 1 insertion(+), 1 deletion(-)
[test1@eliza-PC testrepo]$
```

Push

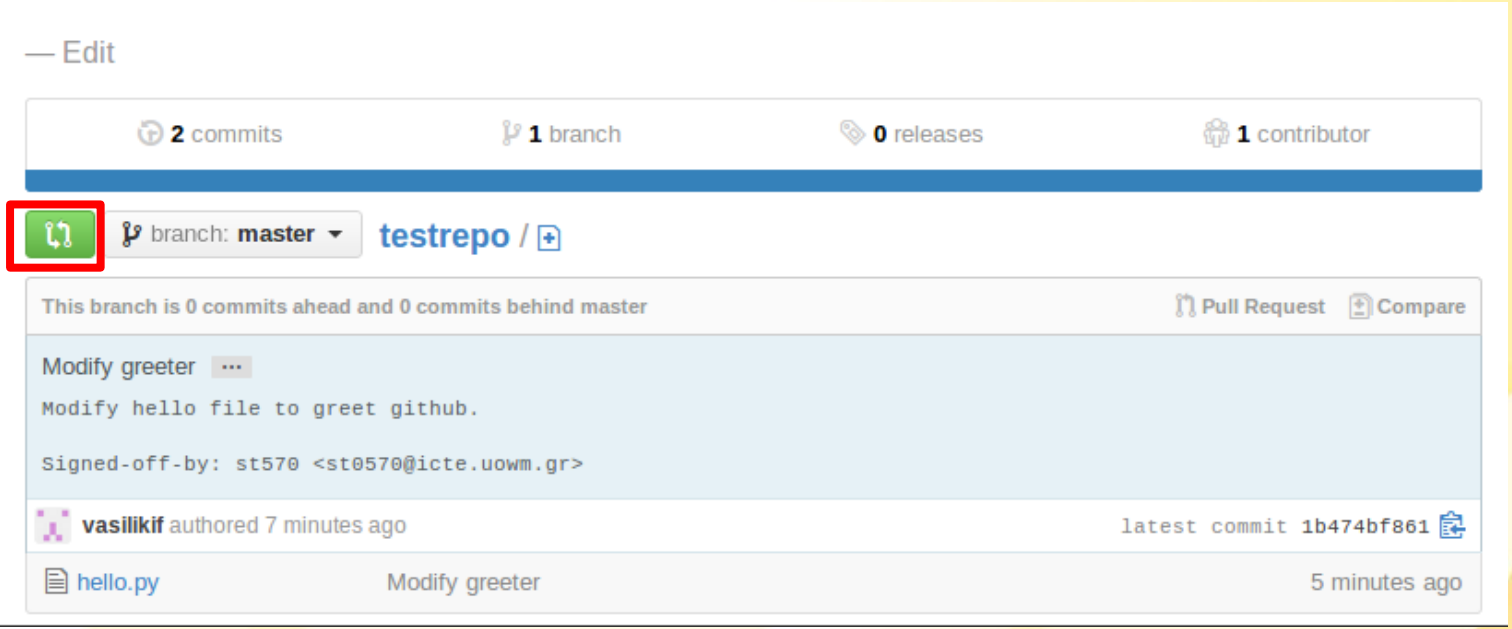
```
[test1@eliza-PC testrepo]$ git log
commit 1b474bf86145fed1e859d4c8fc39b5583e813606
Author: st570 <st0570@icte.uowm.gr>
Date:   Mon May 19 16:18:53 2014 +0300

    Modify greeter

    Modify hello file to greet github.

    Signed-off-by: st570 <st0570@icte.uowm.gr>

commit 3240ce29da17a252cfb721c45f8045f60f030fc9
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Mon May 19 15:39:16 2014 +0300

    Initial commit: Add greeter

    Add hello file which prints "Hello".

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>
[test1@eliza-PC testrepo]$
```

```
[test1@eliza-PC testrepo]$ git remote
origin
[test1@eliza-PC testrepo]$ git branch
* master
[test1@eliza-PC testrepo]$ git push origin master
Username for 'https://github.com': vasilikif
Password for 'https://vasilikif@github.com':
Counting objects: 5, done.
Writing objects: 100% (3/3), 310 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/vasilikif/testrepo.git
   3240ce2..1b474bf  master -> master
```

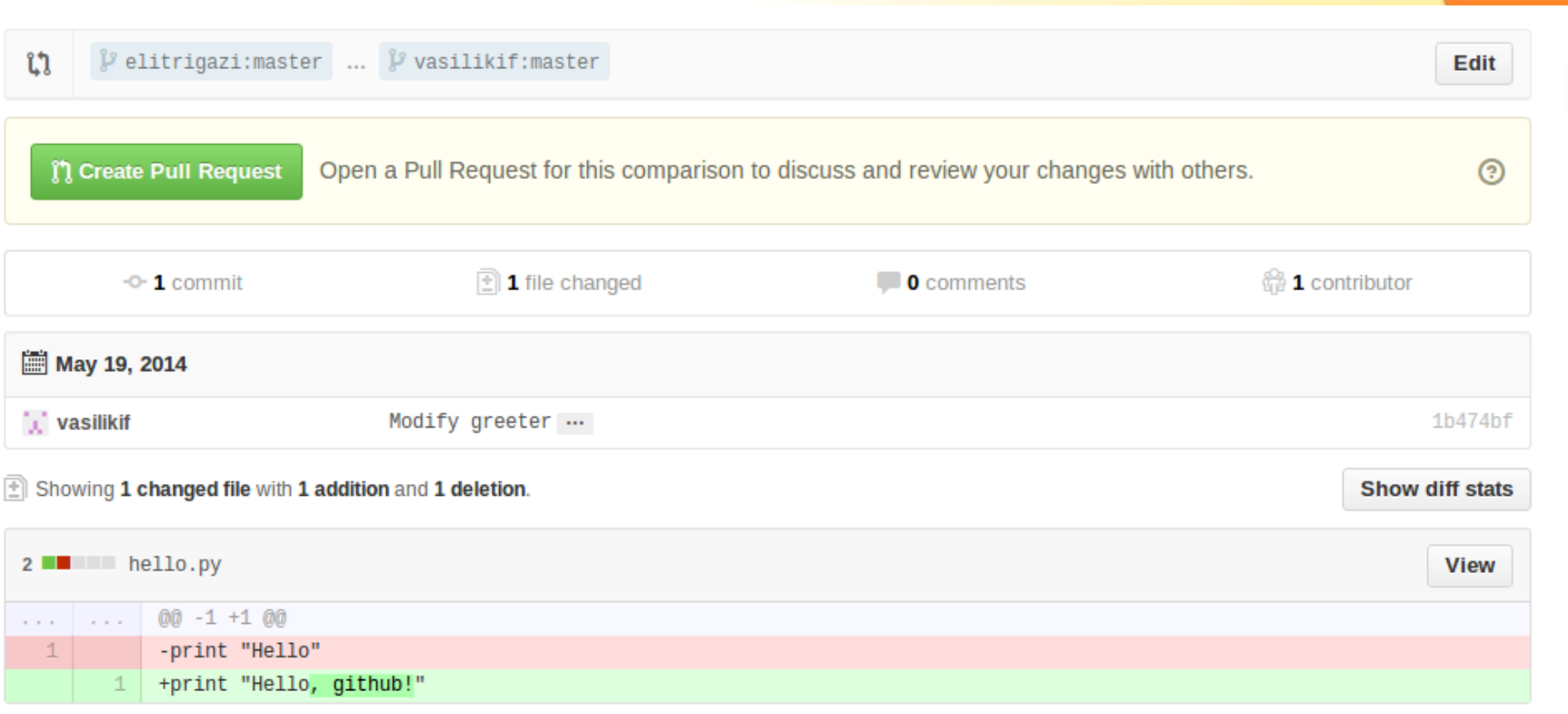
```
[test1@eliza-PC testrepo]$ git remote show origin
* remote origin
  Fetch URL: https://github.com/vasilikif/testrepo.git
  Push  URL: https://github.com/vasilikif/testrepo.git
  HEAD branch: master
  Remote branch:
    master tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (fast-forwardable)
```

Δίνοντας git remote show <remote_name> μπορούμε να δούμε πληροφορίες για ένα απομακρυσμένο αποθετήριο.

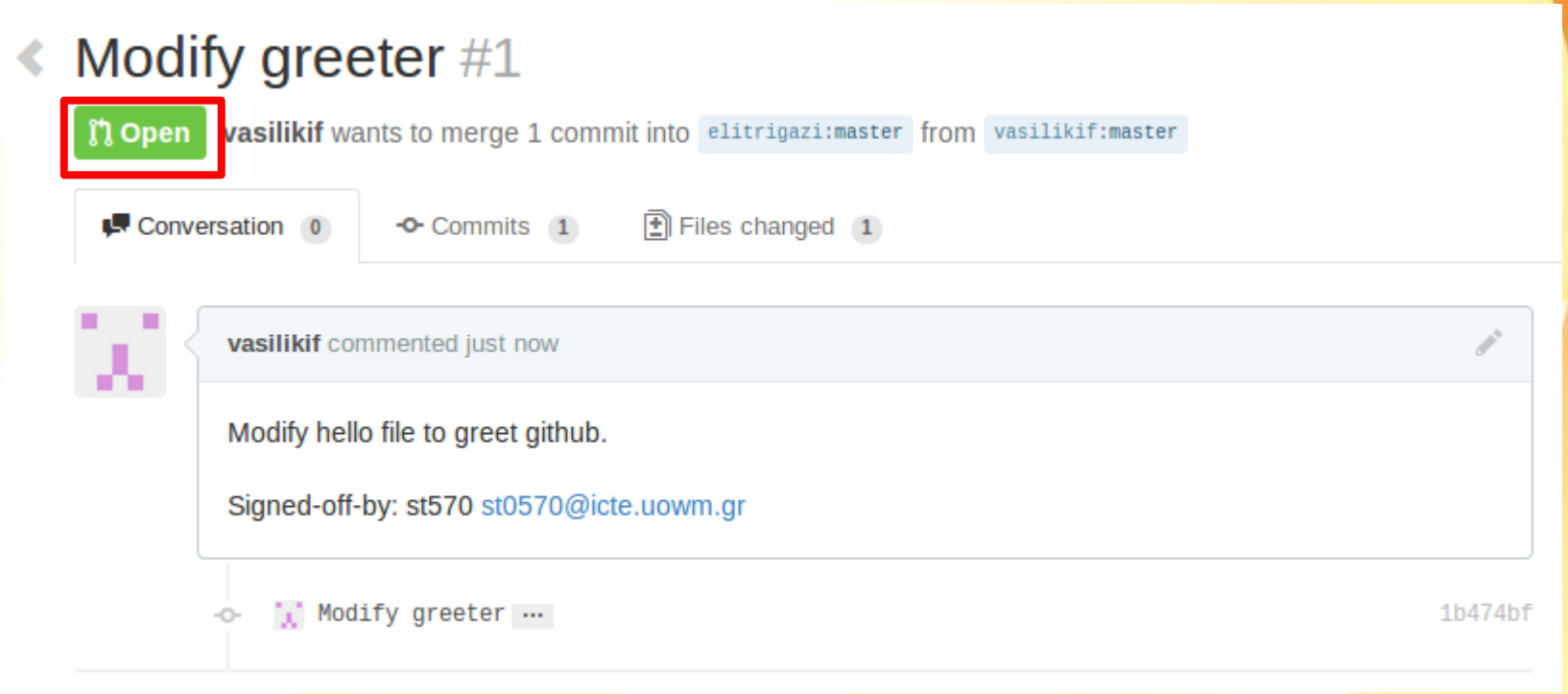
Στο forked project του δεύτερου φοιτητή:



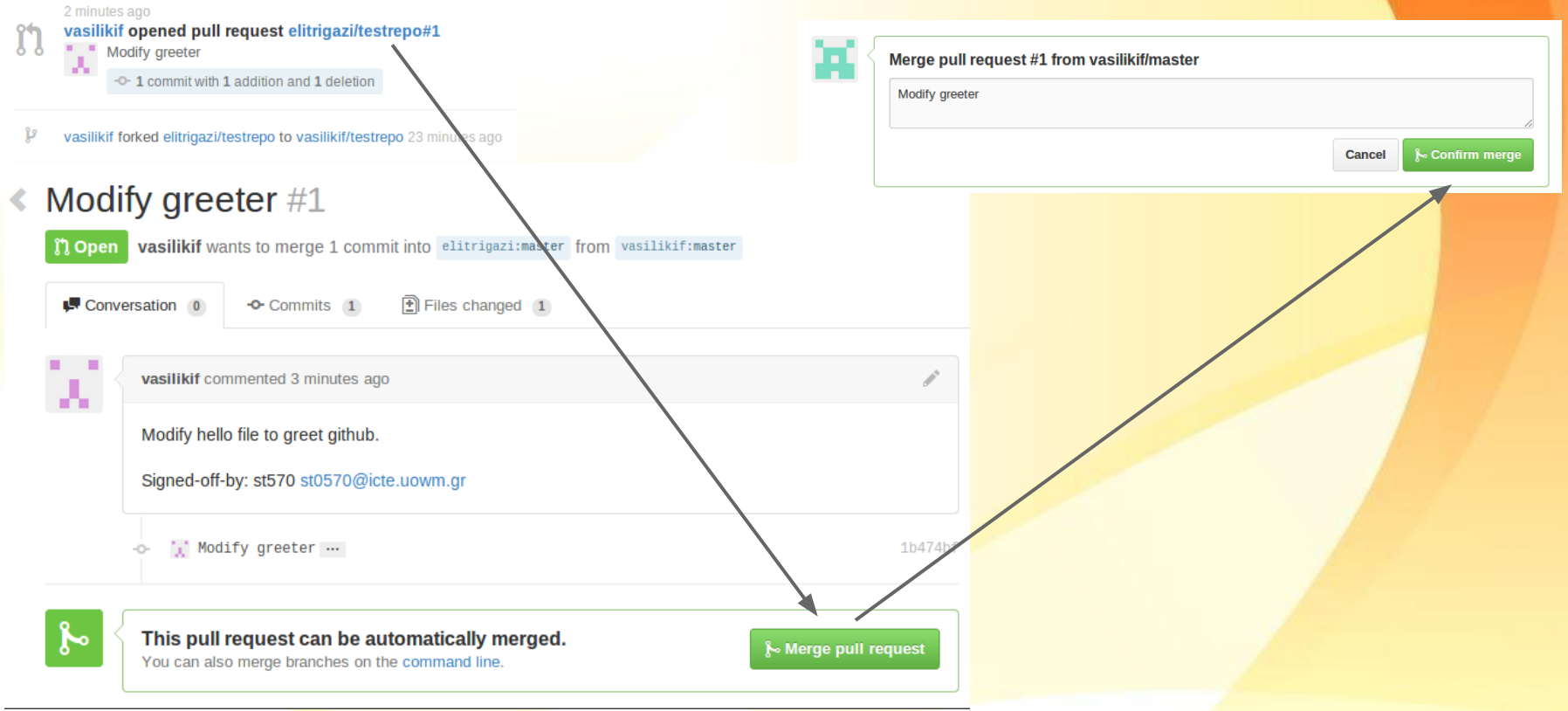
Create pull request



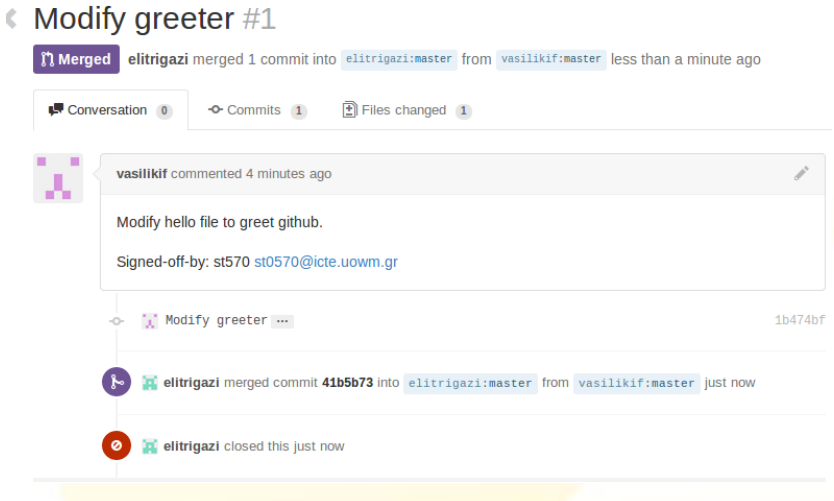
Open pull request

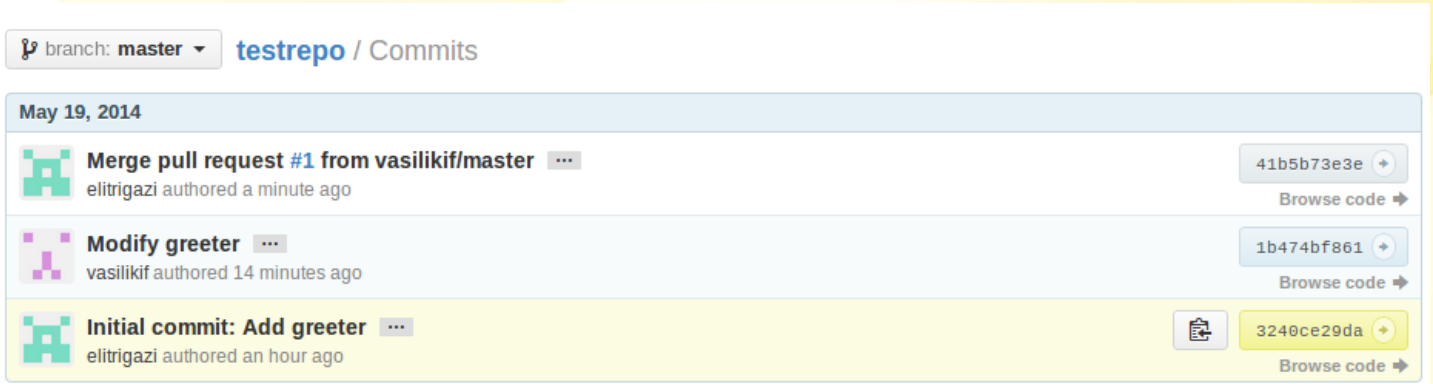


Στον πρώτο φοιτητή:



Review merged request





Pull από τον πρώτο φοιτητή

```
[eliza@eliza-PC testrepo]$ git pull origin master
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 0), reused 4 (delta 0)
Unpacking objects: 100% (4/4), done.
From https://github.com/elitrigazi/testrepo
 * branch            master     -> FETCH_HEAD
Updating 3240ce2..41b5b73
Fast-forward
 hello.py | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
[eliza@eliza-PC testrepo]$ git log
commit 41b5b73e3eb4ebaf6487adf21658ccd6990d8bb7
Merge: 3240ce2 1b474bf
Author: elitrigazi <st0574@icte.uowm.gr>
Date:   Mon May 19 16:31:34 2014 +0300

    Merge pull request #1 from vasilikif/master

    Modify greeter

commit 1b474bf86145fed1e859d4c8fc39b5583e813606
Author: st570 <st0570@icte.uowm.gr>
Date:   Mon May 19 16:18:53 2014 +0300

    Modify greeter

    Modify hello file to greet github.

    Signed-off-by: st570 <st0570@icte.uowm.gr>

commit 3240ce29da17a252cfb721c45f8045f60f030fc9
Author: Elisavet Trigazi <elitrigazi@gmail.com>
Date:   Mon May 19 15:39:16 2014 +0300

    Initial commit: Add greeter

    Add hello file which prints "Hello".

    Signed-off-by: Elisavet Trigazi <elitrigazi@gmail.com>
```

Keep it simple!

Όλη η πολυπλοκότητα στο Git προέρχεται από την αντιμετώπιση των παρακάτω:

Εάν

- βεβαιωθείτε ότι είστε στο τρέχον κεντρικό αποθετήριο
- έχετε κάνει ορισμένες βελτιώσεις στον κώδικα
- έχετε ενημερώσει το κεντρικό αποθετήριο πριν από οποιονδήποτε άλλον

Τότε δεν χρειάζεται να ανησυχείτε για την επίλυση των συγκρούσεων ή την εργασία με πολλαπλά branches.

Επομένως:

- βεβαιωθείτε ότι είστε ενημερωμένοι προτού αρχίσετε να εργάζεστε
- κάνετε commit και ενημερώνετε συχνά το κεντρικό αποθετήριο

Έξτρα εργαλεία

- Για να ενεργοποιήσουμε το auto completion μπορούμε να κάνουμε source το script https://github.com/git/git/blob/master/contrib/completion/git-completion.bash στο .bashrc
- Αν θέλουμε να στήσουμε το δικό μας "github" υπάρχει το project: https://www.gitlab.com/

references

[1] https://www.atlassian.com/git/workflows?_escaped_fragment_=workflow-gitflow#!workflow-gitflow

[2] http://git-scm.com/book/en/Git-Branching-Basic-Branching-and-Merging

Learn more!

1. http://git-scm.com/doc
2. http://readwrite.com/2013/09/30/understanding-github-a-journey-for-beginners-part-1#awesm=~oAVDlqoVghXIkK
3. https://help.github.com/
4. http://ftp.newartisans.com/pub/git.from.bottom.up.pdf