

GitHub

Εγχειρίδιο χρήσης GitHub
μέσω Linux Terminals
ROS-autom 2017[®]

GITHUB

ΠΡΟΕΤΟΙΜΑΣΙΑ ΣΥΣΤΗΜΑΤΟΣ

Επαληθεύουμε ότι έχουμε τα απαραίτητα πακέτα

```
sudo apt-get install git
sudo apt-get install repo
```

Το πιο πιθανό είναι να μην υπάρχει μόνο το repo και να απαιτεί εγκατάσταση. Στη συνέχεια πρέπει να συνδεθεί το github client του υπολογιστή μας, με το github στη σελίδα, ώστε να μπορώ στη συνέχεια να κάνω commits, pulls, pushes κτλ.

Τρέχουμε

```
git config --global user.email dkatikaridis@gmail.com (διπλές παύλες)
git config --global user.name mygithubname
```

Το mail και ο κωδικός πρέπει να είναι σε πλήρη αντιστοιχία με αυτά που βλέπουμε στη σελίδα. τα δικά μου είναι dkatikaridis@gmail.com με username dkat

Στη συνέχεια πρέπει να πιστοποιήσουμε στο github, τον υπολογιστή που θα κάνει push από το λογαριασμό μας.

```
ssh-keygen -t rsa -b 4096 -C dkatikaridis@gmail.com
```

Και θα πάρουμε ως επιστροφή

```
Generating public/private rsa key pair.
Enter file in which to save the key
πατάμε enter
Enter passphrase
πάλι enter
Enter passphrase again
Πάλι enter
Τελικά θα μας δώσει το RSA 4096
```

```
The key's randomart image is:
+---[RSA 4096]-----+
|  +oo+o  .=0+++  |
|  ...oo.=.+.+  |
|  oo+*o. o  |
|  o.+=.  o  |
|  .+. oS.  |
|  . ....E+o.  |
|  o  ...  |
|  .  .  |
|  ....o..  |
+---[SHA256]-----+
```

Στη συνέχεια πρέπει αυτο το ssh key να γραφτεί στον ssh-agent.

```
eval "$(ssh-agent -s)"
```

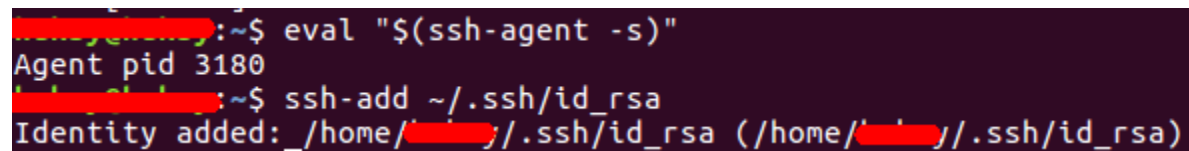
Θα παρουμε μια εξοδο σαν αυτη : **AgentpidXXXXX**

Αποθηκευουμε το ssh σε ενα τοπικο αρχειο

```
ssh-add ~/.ssh/id_rsa
```

Με εξοδο

Identity added : /home/username/.ssh/id_rsa (home/username/.ssh/id_rsa)



```
username:~$ eval "$(ssh-agent -s)"
Agent pid 3180
username:~$ ssh-add ~/.ssh/id_rsa
Identity added: /home/username/.ssh/id_rsa (/home/username/.ssh/id_rsa)
```

Στη συνέχεια πρέπει να αντιγραφουμε το SSHστο clipboard.Ετσι κατεβαζουμε το εργαλειο xclip και αντιγραφουμε το SSHμεσω αυτου,στο clipboard

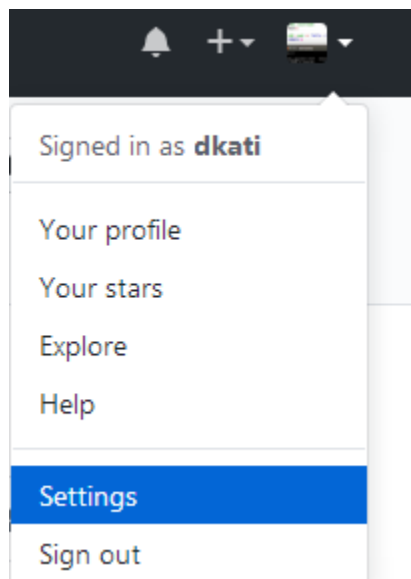
```
sudo apt-get install xclip && xclip -sel clip < ~/.ssh/id_rsa.pub
```

```

dkati@ubuntu:~$ sudo apt-get install xclip
[sudo] password for dkati:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  xclip
0 upgraded, 1 newly installed, 0 to remove and 3 not upgraded.
Need to get 17,0 kB of archives.
After this operation, 72,7 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu xenial/universe amd64 xclip amd64 0.12+svn84-4 [17,0 kB]
Fetched 17,0 kB in 0s (54,0 kB/s)
Selecting previously unselected package xclip.
(Reading database ... 346421 files and directories currently installed.)
Preparing to unpack .../xclip_0.12+svn84-4_amd64.deb ...
Unpacking xclip (0.12+svn84-4) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up xclip (0.12+svn84-4) ...
dkati@ubuntu:~$ xclip -sel clip < ~/.ssh/id_rsa.pub
dkati@ubuntu:~$

```

Στη συνέχεια αυτο το ssh πρεπει να προσθεθει στο λογαριασμο μας στο github. Ανοιγουμε τις ρυθμισεις του λογαριασμου μας



Επιλεγουμε απο τα μενου , SSH and GPG keys και παταμε στο δεξια κουμπι «New SSHkey». Βαζουμε στο Title εναν τιτλο και στο πλαισιο key απλα παταμε δεξι κλικ/επικολληση

Personal settings

Profile

Account

Emails

Notifications

Billing

SSH and GPG keys

Security

Blocked users

Repositories

Organizations

Saved replies

Authorized OAuth Apps

SSH keys

2

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

SSH

Fingerprint:

Added on May 10, 2017 by

Last used within the last 2 weeks

Delete

SSH

Fingerprint:

Added on Jun 6, 2017

Last used within the last 3 days

Delete

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

GPG keys

New GPG key

There are no GPG keys with access to your account.

Το τελικό αποτέλεσμα θα πρέπει να είναι έτσι

5

Title

Key

```
/PHFDF63YNq1ENBQNwezM5B2xMBPZnZueqI1GMFUkmDcqyRjWADCI0sOJ
/uQd4UMsLZkeO3gSYpxQxRmEMtvJVy45tZnot+EXqB2
/j6WnD3Mv3OyWoK53bj8hIH5A0Ny34qCffHEYsE65cDdaNP9nR6pxeAeKG2sH8QUuI+E/efyOYT
/rQEB7zludNyPvCXsogdMg7on7ARjTTMnd9vkXCQZ2QQRaEjYr0LtrZRf3AA6eJD+0fM4nj4j8KNehWziSA
MKgJQV2fk437fGv9qK6+J6r1VyrhKHRmnDbNJHkgybE8chj3OCSWGM6fQN3jeMYtrmKEGI3gDk2S4ztOW
jp6Onn2kMeOqurt+y1PIVdCYefqw6IFRoVIEjmbP4AOhK
/JRgmF0pxnQWUFsfzJWjKlegQWHDH0OFYC8+s4UGgHiLVNsKIF2JstOtVvty7UQ==
dkatkaridis@gmail.com
```

Add SSH key

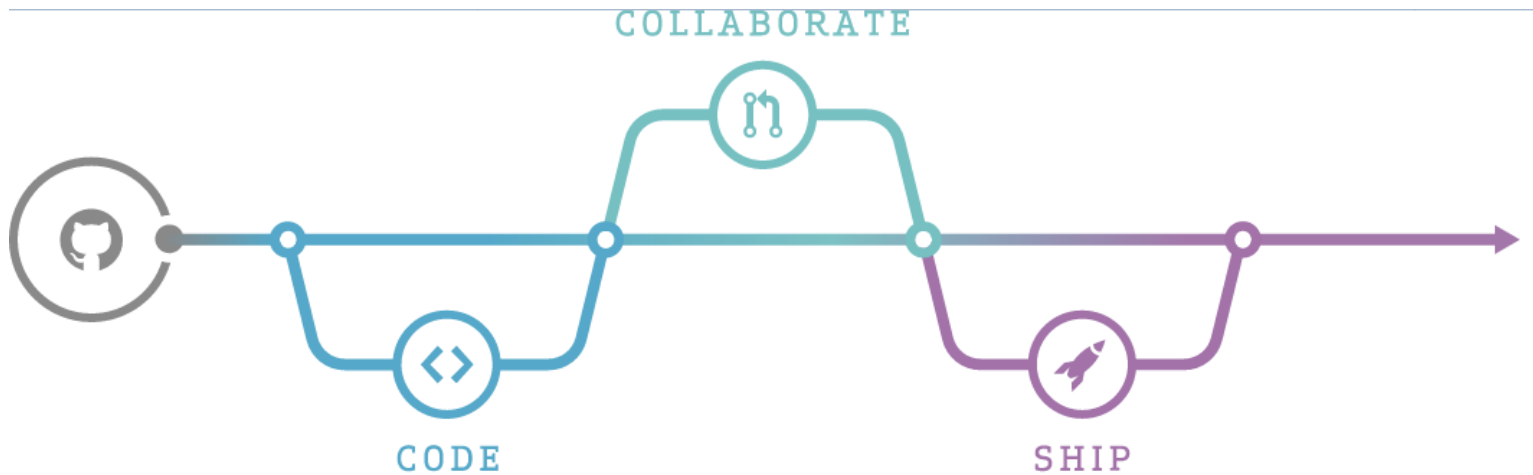
Και πατάμε Add SSHkey.

Ενδεχομενως να ζητησει κωδικο προσβασης.Αν οντως ζητησει απλα τον πληκτρολογουμε.

Πλεον ο υπολογιστης ειναι συνδεμενος απομακρυσμενα με το λογαριασμο μας στο github.

GITHUB

ΤΡΟΠΟΣ ΛΕΙΤΟΥΡΓΙΑΣ ΚΑΙ ΦΙΛΟΣΟΦΙΑ



Τη λειτουργία του github μπορούμε να τη φανταστούμε ως ένα δέντρο (tree) με κλαδιά (branches) και κυριο κορμό (master branch), πάνω σε ένα χρονοδιάγραμμα. Για την ακρίβεια ως ένα δέντρο με κερασία (Θα αναλυθεί παρακάτω). Παρακάτω θα μελετήσουμε ένα flowchart μιας τυχαίας εφαρμογής



Στην εικόνα παρατηρούμε το εξής.

- 1 πλαίσιο με ονομα master
- 1 πλαίσιο με ονομα hotfix
- 1 πλαίσιο release
- 1 πλαίσιο develop
- 2 πλαίσια feature
- Καποια κυκλακια
- 6 διακεκομμενες γραμμες
- Βελακια ροης
- 3 κουτακια που αναφερουν την εκδοση της εφαρμογης καθε φορα

Το κυριο πλαίσιο είναι το master. Είναι το πρωτο branch που δημιουργείται και η κυρια ροή του προγράμματος. Τα υπολοιπα πλαίσια είναι τα δευτερευοντα branches.

Οι διακεκομμενες γραμμες είναι οι οριζοντιες γραμμες που μας δειχνουν την χρονικη εξελιξη των πραγματος απο αριστερα προς τα δεξια.

Τα βελακια ροης δειχνουν τα βηματα που κανει η ροή του github το οποιο θα αναλυσουμε παρακατω.

Τα κυκλακια αντιπροσωπευουν μια αλλαγη(ενα commit) στον κωδικα.

Πολυ σημαντικό είναι να παρατηρησουμε πως τα κυκλακια βρισκονται σε καθετη αντιστοιχια.



Ας το αναλυσουμε αυτο.Στο πρωτο καθετο πλαισιο υπαρχει ΜΟΝΟ το κυκλακι του master branch. Αυτο σημεινει οτι ειναι το ΠΡΩΤΟ-initial release του source code μας στο github.Μπορουμε να φανταστουμε το master branchως το «επισημο» source codeπου θα θελαμε καποιος να δει..Συνηθως ειναι η stable εκδοση του τρεχοντα κωδικα.

Στη συνεχεια παρατηρουμε 3 βελακια.
1 προς το hotfix.1 προς το develop.1 προς το feature.

Αυτο σημεινει οτι απο το master φτιαξαμε αλλα 3 branches.Βλεπουμε πως τα κυκλακια αυτα δεν ειναι στην ιδια καθετο με το master.αυτο σημεινει πως εχει γινει καποιο commit-καποια αλλαγη στον κωδικα. Συνεπως βλεπουμε οτι ο προγραμματιστης του κωδικα εφτιαξε 3 ακομα branch για να μπορει να προσθεσει μια αλλαγη.

-Και γιατι δεν βαζει την αλλαγη κατευθειαν στο master branch?

-Για να μπορει να την ελεγξει.Αν δουλευει σωστα τοτε την προσθετει στο master branch που θα δουμε παρακατω

Στη τριτη καθετη γραμμη βλεπουμε το hotfix να μπαινει στο master.Αυτο σημεινει πως το hot fix ηταν πιθανον ενα bug fix ,οποτε ο προγραμματιστης το προσαρμοσε στο κυριο branch.Επισης ο προγραμματιστης ανεβαζει την εκδοση του προγραμματος σε νο.2

Στην ιδια τριτη καθετη γραμμη παρατηρω και τα αλλα κυκλακια απο develop και feature.

Καθε κυκλακι ειναι και ενα commit(μια αλλαγη) στον κωδικα.Αυτο σημεινει πως αν το κυκλακι που υπαρχει μεσα σε μια καθετη γραμμη ,υπαρχει και σε αλλο branch ,τοτε **το ΙΔΙΟ ακριβως** κομματι κωδικα υπαρχει και σε αυτο το branch.

Με την ιδια λογικη προχωραμε στο διαγραμμα και οπου υπαρχουν βελακια που πανε διαγωνια ,σημεινει πως απο εκεινο το σημειο(commit) ,φτιαχνω νεο branch.Θα τα αναλυσουμε ολα αυτα και παρακατω

Παρατηρησεις

- Το github κραταει ιστορικο των commits .Αν κατι γραφτει στο ιστορικο ,ΔΕΝ ΔΙΑΓΡΑΦΕΤΑΙ.
- Το ιστορικο του github μπορει να αναιρεθει (Να γινει reset) ή να γινει Revert
 - Revert σημεινει να γυρισω τον κωδικα πως ηταν πριν.
Αν πχ,εσβησα ενα declaration μιας μεταβλητης,τοτε με το revert τη δηλώνει ξανά
- Ολοκληρο το source code με τα ολα τα branches ονομαζεται repository (repo)
- Το οτι κανω push μια αλλαγη/ενα commit δεν σημεινει πως αλλαζω branch.Ο λογος που αλλαζω branch ειναι για να μπορω να κανω δοκιμες στον κωδικα χωρις να επιρρεαζω τον βασικο κωδικα που θεωρω stable
- Μεσα σε ενα repository μπορουμε να προσθεσουμε contributors και να εχουν δικαιωμα να αλλαξουν τον κωδικα.Οποτε την επομενη φορα που θα θελω να κανω καποιες αλλαγες,θα πρεπει να «τραβηξω» τις αλλαγες του αλλου contributor

GITHUB

LINUX TERMINAL ΚΑΙ GIT ΕΝΤΟΛΕΣ

Το github αρχικά δημιουργήθηκε με μοναδικό τρόπο χρήσης , τα linux terminals. Αργότερα δημιουργήθηκε και η desktop εφαρμογή για Windows. Εδώ θα αναλυθεί η χρήση μέσω linux terminal. Η εξοικείωση με το github μέσω terminal θα βοηθήσει και αυτόν που θα θέλει να χρησιμοποιήσει την windows desktop εφαρμογή. **Όχι όμως το αντιστρόφω** (Η εφαρμογή για windows περιέχει επίσης terminal , για advanced χρήστες)

Οι τρόποι που μπορώ να ανεβάσω/κατεβάσω τον κώδικά μου στο/από το github είναι δύο.

- Ανεβάσμα/Κατεβάσμα τον κώδικα σε συγκεκριμένο branch , στο github μου, και επεξεργασία. Το μειονέκτημα εδώ είναι πως ορισμένες φορές πρέπει να κατεβαζουμε ΟΛΟΚΛΗΡΟ τον κώδικα από το github και όχι μόνο τα τελευταία commits που κάποιος άλλος contributor έκανε στο project
- Δημιουργία manifest που έχει κατεβασμένα όλα τα branch σε .tmp φάκελο και είναι ΠΑΝΤΑ προσβάσιμα χωρίς πιθανότητα διαγραφής. Το μειονέκτημα εδώ είναι πως ο τρόπος αυτός είναι λίγο πιο περιπλοκός



Παρακάτω θα αναλυθεί ο **δευτερος** τρόπος τον οποίο θεωρούμε πιο ασφαλές. Παρόλα αυτά θα αναφερθούμε και στον πρώτο τρόπο ο οποίος μας βοηθάει στο να παρούμε ολοκληρωτά repository από άλλους

ΞΕΚΙΝΩΝΤΑΣ ΝΕΟ REPOSITORY ΓΙΑ ΝΑ ΑΝΕΒΑΣΟΥΜΕ ΤΟΝ ΚΩΔΙΚΑ ΕΝΟΣ PROJECT ΠΟΥ ΕΧΟΥΜΕ ΓΡΑΨΕΙ

Μεταφερόμαστε στο προφίλ μας στο github.com. Αφού έχουμε ήδη κάνει login , επιλέγουμε το **+** από πάνω δεξιά και επιλέγουμε New repository. Εκεί μπορούμε να συμπληρώσουμε τα στοιχεία του αρχικού repository. Αρχικά θα είναι αδειο και μετά θα κάνουμε προσθήκη του κώδικα μας ως initial release. Γράφουμε το όνομα του repository , μια περιγραφή εάν θέλουμε, Δημοσίο repository και επιλέγουμε και το Initialize this repository with a README για να δημιουργήσει το πρώτο μας αρχείο. Αργότερα θα προσθέσουμε και το license.



Owner

Repository name

 **dkati** ▾ / 

Great repository names are short and memorable. Need inspiration? How about **laughing-bassoon**.

Description (optional)

-
- ☒  **Public**
Anyone can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

| 

Create repository

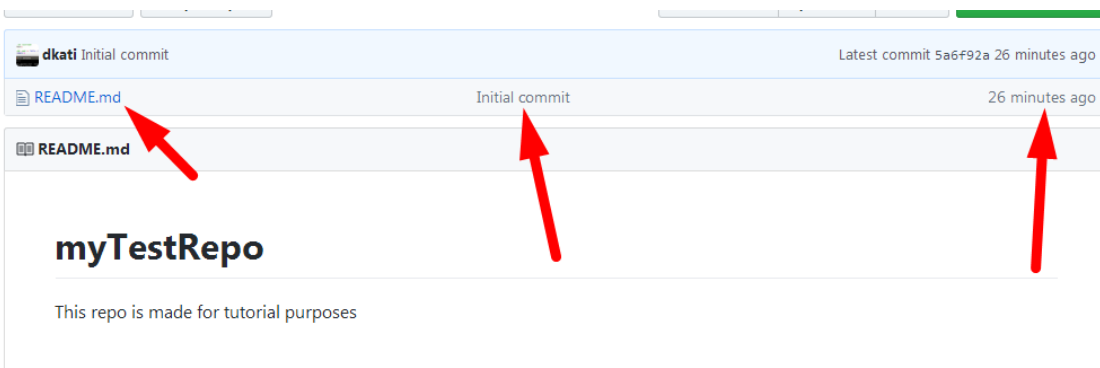
Αφου πατησουμε Create repository θα δουμε την εικονα αυτη

The screenshot shows the GitHub interface for a repository named 'myTestRepo' by user 'dkati'. At the top, there are buttons for 'Unwatch' (1), 'Star' (0), and 'Fork' (0). Below this is a navigation bar with links for 'Code', 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Wiki', 'Settings', and 'Insights'. A message states 'This repo is made for tutorial purposes' with an 'Edit' button. Below this, statistics show '1 commit', '1 branch', '0 releases', and '1 contributor'. Action buttons include 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The commit history shows an 'Initial commit' by 'dkati' with the file 'README.md'. The repository content area shows the 'README.md' file with the title 'myTestRepo' and the same tutorial purpose message.

Παρατηρουμε τα εξης στοιχεια

- Πανω αριστερα βλεπουμε το ονομα του χρηστη και το ονομα του repository.
dkati/myTestRepo
- Απο κατω υπαρχει ενα μενου.
 - Code : Το κυριο και πιο σημαντικο μενου
 - Issues : Στη καρτελα αυτη μπορει οποιοσδηποτε χρηστης (ακομη και αυτος που δεν ειναι contributor) να κανει αναφορα καποιων θεματων-σφαλματων,και οι contributors να τα δουν και να απαντησουν/λυσουν τα ζητηματα
 - Pull requests : Στη καρτελα αυτη ειναι μαζεμενες καποιες προτασεις που καποιος τριτος χρηστης κανει,σχετικα με τον κωδικα.Οι contributors βλεπουν τις αλλαγες και αν θελουν με το πατημα ενος κουμπιου επιτρεπουν το τριτο χρηστη να προσθεσει τα committου στο repository μας,χωρις να ειναι μελος αυτου
 - Wiki : Το παραδοσιακο wikioπου οι contributors δινουν καποιες οδηγιες σχετικα με το source code
 - Settings : Ρυθμισεις του repository(Οχι του source code)

- Παρακατω ειναι το description που εχουμε βαλει στο repository
- Στη συνεχεια υπαρχει αλλο ενα μενου
 - 1 Commit : Πατωντας πανω στο μενου αυτο μας εμφανιζει ολα τα commits που εχουν γινει με χρονολογικη σειρα. Λεπτομερειες για τα commits θα αναφερθουν αργοτερα
 - 1 branch : εμφανιζει τα branches
 - 0 releases : αφορα τα releases που κανουν οι contributors
 - 1 contributor : εμφανιζει ολους οσους συνεισφερει στον κωδικα ειτε ειναι μελη του repository ειτε χρησιμοποιησαμε τον κωδικα του
- Παρακατω βλεπουμε ενα κουμπι-μενου που λεει branch: master. Απο εδω μπορω να αλλαζω branches και να βλεπω τον αντιστοιχο κωδικα και τα αντιστοιχα commits.
- New pull request: Εαν θελω ως τριτος χρηστης να προσθεσω κωδικα
- Create new file/Upload files : Χειροκινητη δημιουργια/δημοσιευση αρχιου (**Δεν συνισταται**)
- Clone/Download : Κατεβασμα του source code σε zip μορφη (Δεν συνισταται! Ενα λογος που δεν συνισταται ειναι οτι στα linux περιβαλλοντα , το zip αρχαιο μπορει να διαγραφει τυχον symlinks κατα το extract
- **dkati** Initial commit : Αναφερεται το τελευταιο commit που εχει γινει. Το github αυτοματα κανει ενα commit οταν δημιουργουμε το repository και προσθετουμε readme. Η μορφη του τιτλου του commit ειναι
<githubusername><Τιτλος commit>
- Ολα τα υπολοιπα απο κατω ειναι τα αρχεια του συγκεκριμενου directory μαζι με τις λεπτομερειες



Το αριστερο βελακι μας δειχνει τα αρχεια που εχει το συγκεκριμενο directory

Το μεσαio βελακι δειχνει το τελευταιο commit που εχει γινει **και εχει επιρρεασει το συγκεκριμενο αρχαιο**

Το δεξια βελακι δειχνει την ωρα που εχει περασει απο το τελευταιο commit που επιρρεασε το αντιστοιχο αρχαιο.

Γενικότερα οι πληροφορίες αυτές μας βοηθούν να έχουμε μια τάξη μέσα σε τεραστίου μεγέθους κωδικές

Δεδομένου λοιπόν ότι καταλαβαμε πως είναι η δομή του repository πάμε να φτιάξουμε **ΑΛΛΟ ΕΝΑ repository με όνομα myTestRepo2** το οποίο θα είναι το 2^ο εργαλείο μας και στη συνέχεια θα εξηγήσουμε και θα φτιάξουμε το manifest που τα συνδέει όλα αυτά μαζί.

Manifest είναι ένα ειδικό repository που περιέχει ένα ή και παραπάνω αρχεία xml τα οποία περιέχουντα github links από τα projects που χρειαζόμαστε για να στησουμε ένα ολοκληρωμένο source code.

Εχουμε παρατηρήσει ότι πολλά source code από επαγγελματικά προγράμματα είναι χωρισμένα σε διάφορα μέρη, που κάθε μέρος είναι ένα εργαλείο. Λόγου χάριν, στο ROS έχουμε διάφορα εργαλεία (Gazebo, ROScore, RVIZ) τα οποία θέλουμε να δουλέουν όλα μαζί.

Ετσι λοιπόν εμείς θα φτιάξουμε εστω 2 repositories για κάθε ένα εργαλείο που θέλουμε να κάνουμε τις δικές μας/custom αλλαγές.

Η δουλειά του manifest είναι να ορίζει στο git service, ποια sources πρέπει να κατεβούν.

Αντί λοιπόν να κατεβαζουμε ένα-ένα τα project μας, τα τοποθετούμε σε ένα directory και το χωρίζουμε σε κομμάτια μέσω του manifest για καλύτερη οργάνωση.

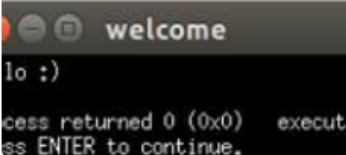
ΔΗΜΙΟΥΡΓΙΑ ΕΝΟΣ MANIFEST

Δημιουργούμε ένα νέο δημόσιο repository με όνομα myproject-manifest, με readme.

Πριν προχωρήσουμε ας δούμε λίγο την αρχική οθόνη του προφίλ μας.

```
include <iostream>
main()

std::cout << "Hello :)
return 0;
```



**Dimitris
Katikaridis**

dkati

[Add a bio](#)

✉ dkatikaridis@gmail.com

- . .

Overview Repositories **41** Stars **0** Followers **15** Following **6**

Search repositories...

Type: **All** ▾

Language: **All** ▾

 **New**

myproject-manifest

Updated 13 seconds ago

myTestRepo2

Updated 6 minutes ago

myTestRepo

This repo is made for tutorial purposes

Updated an hour ago

Παρατηρούμε ότι πλέον έχουμε 3 repositories. Το ένα είναι το myTestRepo που ενδεχομένως να είναι το ένα από τα εργαλεία που source code μου, το δεύτερο είναι το myTestRepo2 το οποίο είναι κάποιο 2 εργαλείο μου και το άλλο είναι το manifest

Πατάμε πάνω στο myproject-manifest και επιλέγουμε από δεξιά «Create new file»
Στην επιλογή ονοματός πληκτρολογούμε «default.xml» και ως περιεχόμενο βάζουμε

```
<?xml version="1.0"encoding="UTF-8"?>
<manifest>

<remote name="github"
        fetch="https://github.com/" />

<project path="myTestRepoLocalDir" name="dkati/myTestRepo" remote="github" revision="master" />
<project path="myTestRepo2LocalDir" name="dkati/myTestRepo2" remote="github" revision="master" />

</manifest>
```

Ας εξηγήσουμε τον κώδικα

Η πρώτη σειρά αφορά την μορφή του XML.

remote είναι το μέλος του xml που ορίζει από που θα τραβεί το git service, όλα τα sources.

το μέλος path του project, είναι το τοπικό directory που θέλουμε να πάρει το κατεβασμένο source code

Το name είναι το github name του repository

Revision είναι το branch το οποίο θα εμφανίζεται στον τοπικό φάκελό μας και το branch το οποίο θα κατεβάζουμε/κάνουμε push.

Κάτω από την επικεφαλίδα Commit new file ορίζουμε το Commit title.

Εδώ ορίζουμε τον τίτλο του commit που θα φαίνεται στα commits. Γενικότερα πρέπει να προσεχούμε τα commit title που κάνουμε καθώς είναι αυτά που κάποιος τρίτος βλέπει. Οπότε η εικόνα που βγαίνει προς τα έξω πρέπει να είναι προσεγμένη. Ένα τυπικό πρότυπο commit title είναι το εξής:

<dir>/<subdir> : <Τι έχω αλλάξει>

Πχ αν έκανα αλλαγές μέσα σε μια class σε ένα αρχείο στο directory mysource/src/libs/mylib.cpp

Τότε στο commit title μπορώ να γράψω ->src/lib: Do not expose _var from mylib

ή κάτι παρόμοιο. Γενικότερα προσπαθούμε να κάνουμε ευστοχά commit titles χωρίς μεγάλη έκταση

Στο description βάζουμε την περιγραφή του commit, αν θέλουμε να εξηγήσουμε κάτι, και πατάμε commit new file

Μετά από τη δημιουργία του αρχείου μπαίνουμε στο αρχείο readme.md και επιλέγουμε το δεξιά μολύβι ώστε να το επεξεργαστούμε. Μέσα στο αρχείο θα γράψουμε την κύρια εντολή που μας κατεβάζει το manifest στον υπολογιστή και το ρυθμίζει. Η εντολή είναι η


```
repo init -u git://github.com/dkati/myproject-manifest.git -b master
```



Στη συνέχεια πρέπει να συνταξουμε ένα commit title.


Ενας αποδεκτος τιτλος θα μπορουσε να ειναι ->**readme:Add our repo init command**

Αφου κανουμε το commit επιστρεφουμε στο myproject-manifest

Παρατηρουμε οτι το επεξεργασμενο readme φαινεται μπροστα στο repository.**Αυτο ισχυει μονο για τα αρχεια readme.**Δεν σημαινει πως οποιαδηποτε αλλαγη σε αρχειο θα εμφανιζεται μπροστα.

 **dkati** committed on **GitHub** readme:Add our repo init command Latest commit 7eb7f47 an hour ago


| | | |
|---|----------------------------------|-------------|
|  README.md | readme:Add our repo init command | an hour ago |
|  default.xml | Create default.xml | an hour ago |

 **README.md**

myproject-manifest

Get our manifest

```
repo init -u git://github.com/dkati/myproject-manifest.git -b master
```

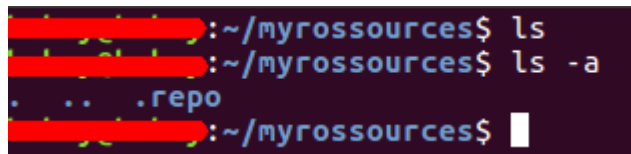


ΑΡΧΙΚΟΠΟΙΗΣΗ ΤΟΥ MANIFEST ΚΑΙ ΤΩΝ SOURCECODES

Απο τη στιγμή που ετοιμάσαμε τα repositories και το manifest, μπορούμε πλέον να ξεκινήσουμε τη διαδικασία της αποθήκευσης των repositories αυτών, στον υπολογιστή μας. Εκτελούμε

```
cd  
mkdir myrossources && cd myrossources
```

Αν μας βγάλει μήνυμα σχετικά με τα χρώματα του λογαριασμού πατάμε 'Y' και προχωράμε. Με μια πρώτη ματιά μέσα στο φάκελο δεν φαίνεται να υπάρχουν αρχεία. Παρόλα αυτά με την εντολή `ls -a` παρατηρούμε ότι υπάρχει ένα νέος φάκελος `.repo`



```
~/myrossources$ ls  
~/myrossources$ ls -a  
.  
..  
.repo  
~/myrossources$
```

Στο φάκελο αυτό υπάρχει το symlink `manifest.xml` το οποίο δείχνει στο `.repo/manifests/default.xml` το οποίο είναι το manifest που γράψαμε πριν.

Ο φάκελος `repo` έχει μέσα τα απαραίτητα αρχεία για να δουλέψει το `git/reposervice` και δεν θα ασχοληθούμε με αυτά.

Πηγαίνουμε λοιπόν ένα directory πίσω και εκτελούμε

```
reposync
```

ΠΡΟΣΟΧΗ.

Το `repo sync` εκτελείται παντοτε στο ίδιο directory με τον φάκελο `.repo`

Αν καταλαθώς εκτελεστεί η εντολή μέσα σε κάποιον φάκελο, θα υπάρχει πρόβλημα σε όλο το `source`

```

[redacted]~/myrossources$ ls
[redacted]~/myrossources$ ls -a
.  ..  .repo
[redacted]~/myrossources$ cd .repo
[redacted]~/myrossources/.repo$ ls
manifests manifests.git manifest.xml repo
[redacted]~/myrossources/.repo$ cd ..
[redacted]~/myrossources$ repo sync

... A new repo command ( 1.23) is available.
... You should upgrade soon:

cp /home/[redacted]/myrossources/.repo/repo/repo /usr/bin/repo

Fetching project dkati/myTestRepo
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           % Dload  Upload    Total     Spent    Left     Speed
0         0     0     0     0     0     0     0  --:--:-- --:--:-- --:--:--    0
curl: (22) The requested URL returned error: 404 Not Found
Server does not provide clone.bundle; ignoring.
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
From https://github.com/dkati/myTestRepo
* [new branch]      master    -> github/master
Fetching projects: 50% (1/2)  Fetching project dkati/myTestRepo2
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           % Dload  Upload    Total     Spent    Left     Speed
0         0     0     0     0     0     0     0  --:--:-- --:--:-- --:--:--    0
curl: (22) The requested URL returned error: 404 Not Found
Server does not provide clone.bundle; ignoring.
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
From https://github.com/dkati/myTestRepo2
* [new branch]      master    -> github/master
Fetching projects: 100% (2/2), done.

[redacted]~/myrossources$

```

Μετα απο τη τελευταια μας εντολη , εχουμε κατεβασει ολο το source code στον υπολογιστη μας.

Μπορουμε επισης να κατεβασουμε μονο το 1 repository εαν θελουμε κανοντας

repo sync myTestRepoLocalDir

Αυτο θα κανει μια ανανεωση το τοπικο source code με αυτο του github.Αυτο βοηθαει οταν θελουμε να αναιρεσουμε τις αλλαγες μας.

Παντοτε η λογικη ειναι να εχουμε updated τα source στο github ,ωστε να μπορουμε να κανουμε αλλαγες ,τοπικα και με ασφαλεια.Εαν λοιπον κατι παει στραβα,απλα σβηνω τον φακελο και εκτελω repo sync για το repository που θελω

Το repo sync μπορει να παρει ως ορισμα τα threads με τα οποια θα κανει sync.(-jX δηλωνει το πληθος των ταυτόχρονων διαδικασιών που θέλουμε να γίνονται)

repo sync -j4

Γιατι ομως να μας ενδιαφερει το ποσα threads τρεχουν κατα το repo sync απο τη στιγμη που γινονται ολα μεσω ιντερνετ ?

Γιατι το repo ,κατεβαζει το source code σε συμπιεσμενη μορφη,αποσυμπιεζει τα αρχεια και μεσω μιας διαδικασιας diff check ελεγχει ποια αρχεια απο το source μας εχουν αλλαξει ωστε να τα αντικαταστησει.

Αν εχουμε προβλημα με το μεγαλυτερο -j απλα το μειωνουμε.

Σημειωση , το -j επιρρεαζει και τις εργασιες που κανει το δικτυο κατα το κατεβασμα.Πρακτικα ,μεγαλυτερο j σημαινει πιο γρηγορο κατεβασμα με μεγαλυτερη πιθανοτητα σφαλματος.Αυτο διαφοροποιειται απο συνδεση σε συνδεση διαδικτυου

Μεχρι το σημειο αυτο,ολα πρεπει να γινουν μια και μονο φορα.Οτι ακολουθει ειναι αυτα που πρεπει να γνωριζουμε ωστε να εξοικιωθουμε με το github και τον τροπο λειτουργιας του.

ΤΟ ΠΡΩΤΟ COMMIT ΚΑΙ ΤΟ ΠΡΩΤΟ PUSH

Το πιο σημαντικο σημειο ειναι να καταλαβουμε τι θελουμε να κανουμε push και ποτε θελουμε να το κανουμε push.Οπως ειπαμε και πριν ενα commit πρεπει να ειναι καθαρο.Αυτο σημαινει οτι το καθε commit πρεπει

- Να εχει ξεκαθαρο τιτλο που θα περιγραφει τι ακριβως κανει
- Αν ειναι απαραιτητο να εχει μια περιγραφη με λεπτομεριες
- Να μην περιεχει αλλαγες σε αρχεια που δεν αφορουν την κυρια αλλαγη και σκοπο του commit.
- Να μην ειναι αντιγραφη απο αλλο commit.Αν θελω το commit καποιου τριτου θα το κανω με τον νομιμο τροπο που θα δουμε παρακατω

Ας εξηγησουμε την 3^η περιπτωση.

Κατα την εξοικιωση μας με το github,κανουμε pushπραγματα τα οποια δεν πρεπει να γινουν push.

Εστω οτι θελουμε να κανουμε push μια αλλαγη σε ενα φακελο που περιεχει 3 αρχεια με σκοπο να γινει κατι συγκεκριμενο.Καλο θα ειναι λοιπον να μην αλλαξουμε κανενα αλλο αρχειο που δεν αφορα την κυρια επεξεργασια.

Παραδειγμα

Σε ενα αρχειο αλλαζω 2 μεταβλητες απο int σε double και σε ενα αλλο αρχειο αλλαζω το ονομα μιας class.

Στο commit title βαζω "src:mylibSwitch 2 vars from int to double".Παρολαυτα υπαρχει και η αλλαγη του ονοματος της class και θα γινει και αυτο push.Κατι το οποιο δεν θα θελαμε.

Πηγαινουμε λοιπον να κανουμε την πρωτη μας αλλαγη ,εστω στο myTestRepo
Ο τοπικος φακελος του repository αυτου,ειναι το myTestRepoLocalDir.Οποτε

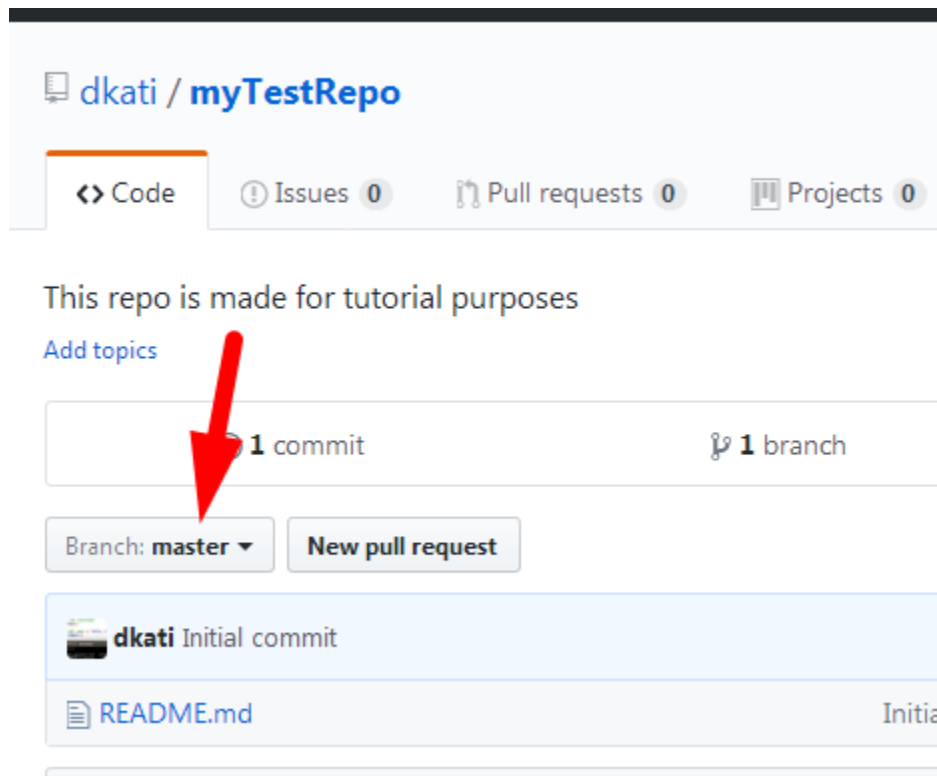
Cd myTestRepoLocalDir

Αφου μπηκαμε στο directory πρεπει να δηλωσουμε το branch στο οποιο ειμαστε.Αρχικα να δουμε αν κατα τυχη ειμαστε ηδη σε branch.Αυτο μπορεί να προκληθει απο καποια παλιότερη μας διαδικασια μεσα στο φακελο.Εκτελουμε

git branch

Και λογικα μας εμφανιζει (nobranch)

Αυτο σημαινει οτι δεν ειμαστε σε κανενα branch.Βλεπουμε απο τη σελιδα του repo μας,οτι το branch μας λεγεται master.



Αν θελω να στείλω τις μελλοντικές μου αλλαγές στο branch αυτο τότε εκτελω

```
git branch master
```

```
git checkout master
```

Ή πιο απλά

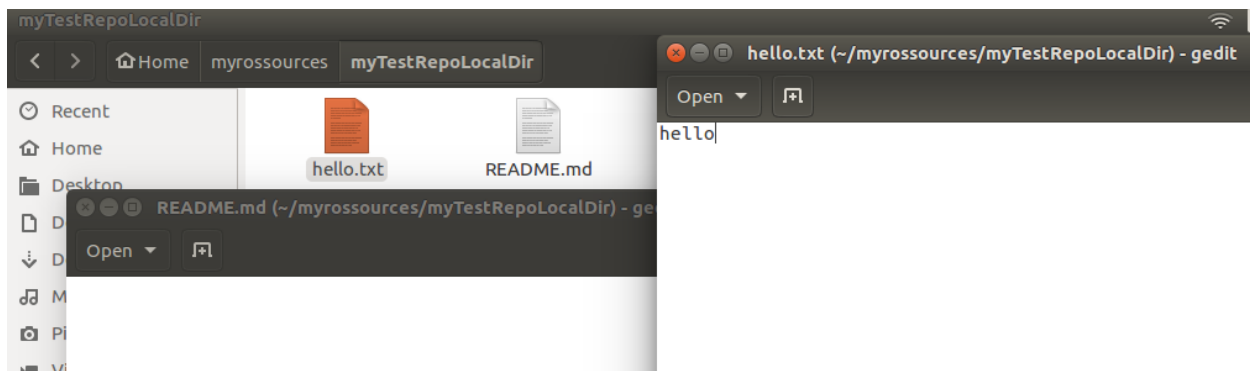
```
git checkout -b master
```

```
myrossources/myTestRepoLocalDir$ git branch
* (no branch)
myrossources/myTestRepoLocalDir$ git branch master
myrossources/myTestRepoLocalDir$ git checkout master
Switched to branch 'master'
myrossources/myTestRepoLocalDir$
```

Απο τη στιγμή που άλλαξα το branch μου, μπορω είτε να κρατησω το τερματικο ανοιχτο είτε να το κλεισω. Δεν υπαρχει κατι που τρεχει στο background.

Κανω λοιπον τις αλλαγες μου τοπικα.

Θα προσθεσουμε ενα αρχειο με ονομα hello.txt, μεσα θα βαλουμε τη λεξη "Hello" και θα σβησουμε απο το readme οτι περιεχει



Εστω οτι αυτη ειναι η αλλαγη που θελαμε να κανουμε. Τωρα μενει να το προσθεσουμε στο commit.

git add -A
git commit

Αυτοματως θα μας εμφανισει ενα παραθυρο του nano οπου κανουμε επεξεργασια το commit μας. Η πρωτη σειρα ειναι παντα το commit title. Αφηνοντας μια σειρα και προσθετοντας μια τριτη, γραφουμε την περιγραφη.

```
My first commit!
thats my first commit!YaY!
# Please enter the commit message for your
# with '#' will be ignored, and an empty me
# On branch master
# Changes to be committed:
#   modified:   README.md
#   new file:   hello.txt
#
```

Παρατηρουμε τις δυο σειρες

modified: README.md

new file: hello.txt

Αυτο μας επαληθευει οτι δεν «τραβηξαμε» στο commit καποιο αρχειο που δεν θα επρεπε να ειναι μεσα.

Αν ειναι ολα σωστα, τοτε συνεχιζουμε.

Διαφορετικα παταμε ctrl+x και μετα Υ χωρις να εχουμε συμπληρωσει commit title και description για να βγουμε. Πηγαινουμε στο φακελο που ειναι το .repo (δηλαδη στο root του myrossources) και ειτε διαγραφουμε το φακελο και κανουμε παλι repo sync ειτε κανουμε κανονικα το push και μετα κανουμε νεο commit για το fix(που δεν συστηνεται). Μια αλλη τεχνικη ειναι να κρατησουμε τα αρχεια που καναμε επεξεργασια , σε εναν αλλο φακελο, να κανουμε το repo sync και μετα paste τα αρχεια που ειναι επεξεργασμενα

Παταμε ctrl+X και Y για να κλείσει το commit. Βλεπουμε στο τερματικο οτι δειχνει το τιτλο του commit και αναφερει οτι 2 αρχεια αλλαξαν, 1 προσθηκη και 2 διαγραφες. Οι προσθηκες και οι διαγραφες αφορουν ΣΕΙΠΕΣ κωδικα. Επισης μας δειχνει οτι δημιουργηθηκε το hello.txt με mod 0644

Απο τη στιγμη που εγινε το commit, πρεπει να το κανουμε push.

Git push origin master

Η εντολη αυτη θα προσπαθησει να στείλει το commit στο branch master.

Παρολαυτα μας βγαζει καποια fatal errors. Ο λογος ειναι οτι χρειαζεται να φτιαξουμε (απαιτειται μονο μια φορα) το remote. Το remote ειναι υπευθυνο για την αποστολη του τοπικου commit, στο προφιλ μας. Συνεπως εκτελουμε (μια φορα και μονο φορα)

git remote add origin

Και ξανα κανουμε **git push origin master**

Θα μας ρωτησει αν θελουμε πραγματι να γινει το push (Θα ρωτησει μονο για μια φορα και ποτε ξανα). παταμε yes και το στελνει. Αν βλεπουμε την παρακατω οθονη, τοτε ολα εγιναν σωστα

```
~/myrossources/myTestRepoLocalDir$ git push origin master
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
~/myrossources/myTestRepoLocalDir$ git remote add origin git@github.
com:dkati/myTestRepo.git
~/myrossources/myTestRepoLocalDir$ git push origin master
The authenticity of host 'github.com (192.30.253.112)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUPJWG17E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,192.30.253.112' (RSA) to the list of know
n hosts.
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 305 bytes | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To git@github.com:dkati/myTestRepo.git
  5a6f92a..f8d94ec master -> master
```

Στα commits λοιπον του project ,πλεον βλεπω την αλλαγη μου

https://github.com/dkati/myTestRepo/commits/master

Αναζήτηση

This repository Search Pull requests Issues Marketplace Gist

dkati / myTestRepo

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

Branch: master

Commits on Jun 9, 2017

My first commit! f8d94ec

dkati committed 18 minutes ago

Initial commit 5a6f92a

dkati committed 5 hours ago

Κανοντας κλικ πανω στο commit βλεπω τις λεπτομεριες.

https://github.com/dkati/myTestRepo/commit/f8d94ec598e8e94ccfb5c61ca0c5c973ca61cc1e

Αναζήτηση

This repository Search Pull requests Issues Marketplace Gist

dkati / myTestRepo

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

My first commit! Browse files

thats my first commit!YaY!

master

dkati committed 19 minutes ago 1 parent 5a6f92a commit f8d94ec598e8e94ccfb5c61ca0c5c973ca61cc1e

Showing 2 changed files with 1 addition and 2 deletions. Unified Split

2 README.md

@@ -1,2 +0,0 @@

1 -# myTestRepo

2 -This repo is made for tutorial purposes

1 hello.txt

@@ -0,0 +1 @@

1 +hello

0 comments on commit f8d94ec

Lock conversation

Παρατηρούμε όλα τα στοιχεία του commit. Το τίτλο, τις αλλαγές μας και την περιγραφή.

Οι κοκκινές γραμμές είναι ότι σβήστηκε από το αρχείο ενώ οι πράσινες είναι αυτές που έχουν προστεθεί. Αν όλες οι γραμμές από ένα αρχείο είναι πράσινες, σημαίνει πως το αρχείο έχει δημιουργηθεί κατά το commit.

ΑΝΤΙΣΤΡΟΦΗ ΕΝΟΣ COMMIT

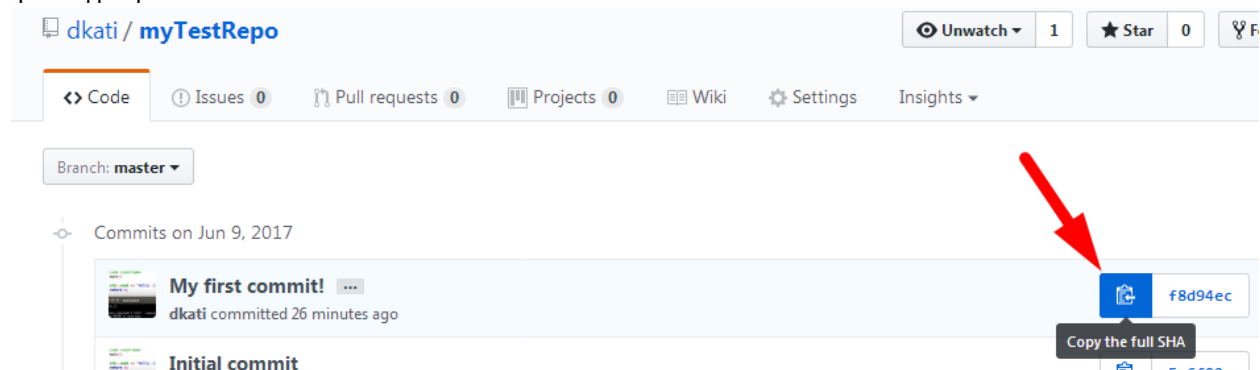
Εχουμε ήδη αναφέρει πως στο github δεν μπορούμε να σβήσουμε commits. Οτι κάνουμε push παραμένει στο ιστορικό. Παρόλαυτα μπορούμε να αντιστρέψουμε ένα commit και αυτό γίνεται με μια μόνο εντολή.

git revert SHA_CODE

Το SHA είναι το μοναδικό κλειδί που έχει κάθε commit και το βρίσκουμε μέσα από το commit ή από το ιστορικό.



Οποτε είτε το παίρνουμε με copy paste από εκεί είτε με ένα απλό κλικ, όπως φαίνεται στην κάτω φωτογραφία



Αυτό θα μας αποθηκεύσει στο clipboard, το commit SHA

Αρα τρεχουμε `git revert fb76ff5c019adfo4bf2369583fbcee346a1c4dd`

Οταν κανουμε το revert μας ανοιγει και παλι το commit message.

Συνηθιζεται να αφηνουμε το commit title οπως ειναι,και να αλλαζουμε το description εξηγωντας γιατι το καναμε αντιστροφη.

αφου κανουμε τις απαραιτητες αλλαγες βγαινουμε μαζι με ctrl+x και y

```
Revert "My first commit!"

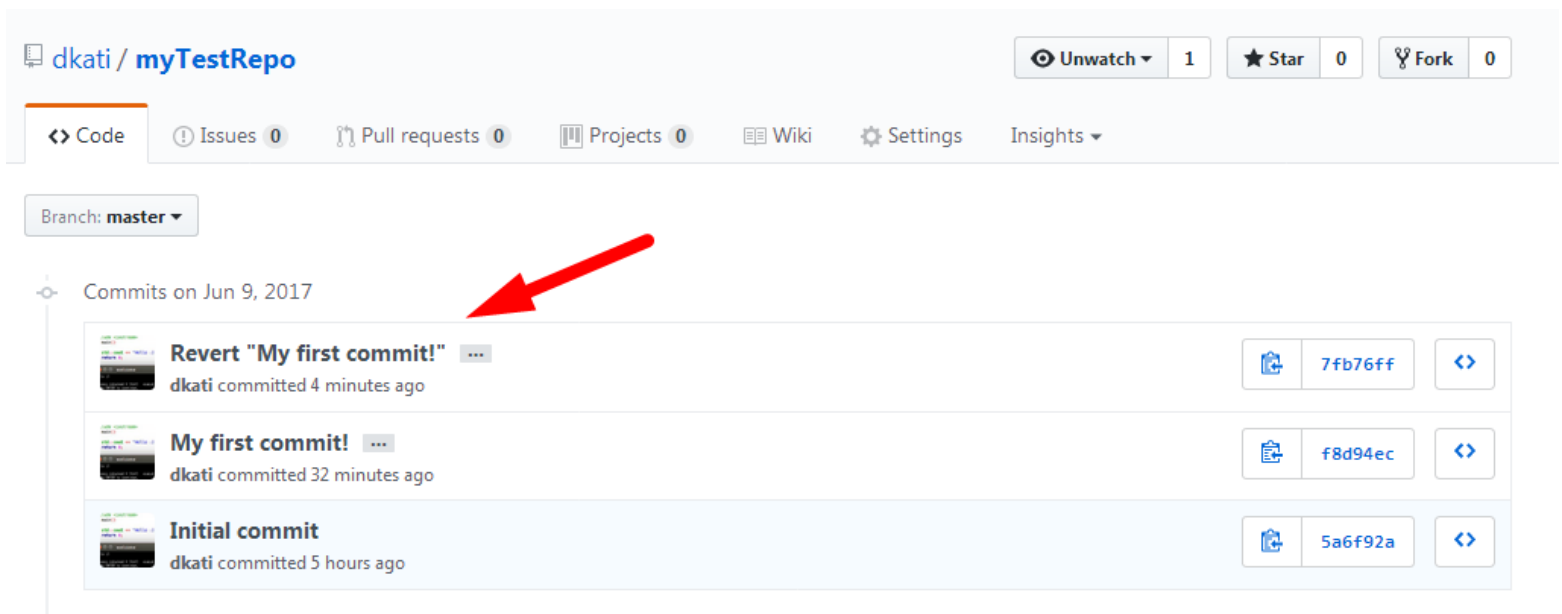
i believe its not needed.breaks the build
This reverts commit f8d94ec598e8e94ccfb5c61ca0c5c973ca61cc1e.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#   modified:   README.md
#   deleted:    hello.txt
#
```

Και κανουμε το push

`git push origin master`

Αφου ολοκληρωθει το push ,μπορω να δω το commit στο commit history



dkati / myTestRepo

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

Branch: master

Commits on Jun 9, 2017

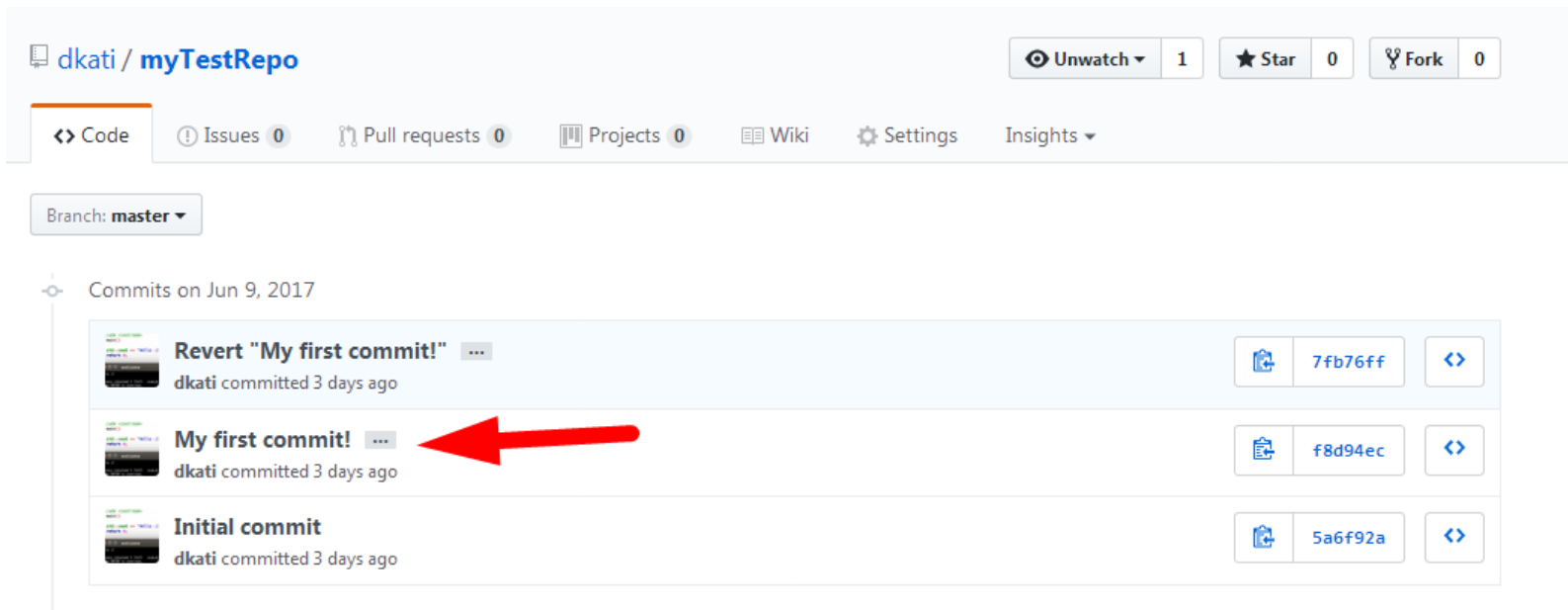
| Commit Message | Author | Time Ago | Hash | View |
|---------------------------|--------|----------------|---------|----------------------|
| Revert "My first commit!" | dkati | 4 minutes ago | 7fb76ff | View |
| My first commit! | dkati | 32 minutes ago | f8d94ec | View |
| Initial commit | dkati | 5 hours ago | 5a6f92a | View |

CHERRY-PICKING.Η «NOMIMH» ΑΝΤΙΓΡΑΦΗ ΚΩΔΙΚΑ

Οποιοσδήποτε κωδικας στο github μπορεί να ασφαλιστει με καποιο διεθνες license(apache,MIT κτλπ). Τα license προστατευουν τον συντακτη απο τυχον «κλοπες» του κωδικα. Το github εχει μεριμνησει για την εξασφαλιση της νομιμοτητας μεσω των λειτουργιων του,και δινει τη δυνατοτητα στο χρηστη να αντιγραψει ενα commit καποιου τριτου προγραμματιστη ,δινοντας τα απαραιτητα credits. Το cherry-pick μας βοηθαει στο να ανανεωνουμε κομματια κωδικα απο κωδικες αλλων προγραμματιστων,δινοντας παντα αυτοματως τα απαραιτητα credits.

Για να μπορεσουμε να κανουμε cherry-pick,πρεπει πρωτα να κατεβασουμε ΟΛΟΚΛΗΡΟ του source code,μαζί με τα commits,απο το οποιο θελουμε καποια συγκεκριμενα commits.

Εστω οτι θελουμε να κανουμε cherry-pick το commit που καναμε πιο πριν "myfirstcommit" απο το <https://github.com/dkati/myTestRepo>



The screenshot shows the GitHub interface for the repository `dkati / myTestRepo`. At the top, there are buttons for `Unwatch` (1), `Star` (0), and `Fork` (0). Below this is a navigation bar with links for `Code`, `Issues` (0), `Pull requests` (0), `Projects` (0), `Wiki`, `Settings`, and `Insights`. A dropdown menu shows the current branch is `master`. The commit history for June 9, 2017, is displayed. The commits are:

- `Revert "My first commit!"` (commit hash `7fb76ff`)
- `My first commit!` (commit hash `f8d94ec`) - This commit is highlighted with a red arrow.
- `Initial commit` (commit hash `5a6f92a`)

Αφου επιλεξαμε το commit που θελουμε πρεπει να παμε μεσω του τερματικου στο directory του project που θελω να βαλω το commit.Εστω οτι θελουμε να μπει στο myTestRepo2 LocalDir.Εκτελουμε

```
git fetch https://github.com/dkati/myTestRepo
```

Η συνταξη της εντολης ειναι
`git fetch<link του προτζεκτ που θελουμε>` (βλ. Παραρτημα)

Με την παραπάνω εντολή κρατήσαμε το ιστορικό των commits του συγκεκριμένου repository.

```
~/myrossources/myTestRepo2LocalDir$ git fetch https://github.com/dkati/mytestrepo master
warning: no common commits
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 7 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
From https://github.com/dkati/mytestrepo
* branch      master      -> FETCH_HEAD
~/myrossources/myTestRepo2LocalDir$
```

Πλέον ειμαστε ετοιμοι να «τραβήξουμε» το commit

```
git cherry-pick f8d94ec598e8e94ccfb5c61ca0c5c973ca61cc1e
```

Η συνταξη της εντολής είναι

```
git cherry-pick <SHA code>
```

Τρεχοντας την εντολή μας βγαζει error που αναφερει οτι δεν μπορεί να προσαρμοσει το commit μέσα στον κωδικα μας.

Όταν κανουμε cherry-pick,κατι τετοιο είναι πολυ συχνο και οφειλουμε να ειμαστε σε θέση να το διορθωσουμε.Το αποτελεσμα της εντολής είναι το παρακατω

```
~/myrossources/myTestRepo2LocalDir$ git cherry-pick f8d94ec598e8e94ccfb5c61ca0c5c973ca61cc1e
error: could not apply f8d94ec... My first commit!
hint: after resolving the conflicts, mark the corrected paths
hint: with 'git add <paths>' or 'git rm <paths>'
hint: and commit the result with 'git commit'
Recorded preimage for 'README.md'
~/myrossources/myTestRepo2LocalDir$
```

Κοιτώντας το commit παρατηρούμε ότι οι αλλαγές είναι σε 2 αρχεία.

https://github.com/dkati/myTestRepo/commit/f8d94ec598e8e94ccfb5c61ca0c5c973ca61cc1e

Αναζήτηση

This repository Search Pull requests Issues Marketplace Gist

dkati / myTestRepo Unwatch 1 Star 0 Fork 0

<> Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

My first commit!

thats my first commit!YaY!

master

dkati committed 19 minutes ago 1 parent 5a6f92a commit f8d94ec598e8e94ccfb5c61ca0c5c973ca61cc1e

Showing 2 changed files with 1 addition and 2 deletions. Unified Split

2 README.md

@@ -1,2 +0,0 @@

1 -# myTestRepo

2 -This repo is made for tutorial purposes

1 hello.txt

@@ -0,0 +1 @@

1 +hello

0 comments on commit f8d94ec Lock conversation

Παρολαυτα στο τερματικο μας βγαλε

Recorded preimage for README.md(βλ.παραρτημα*).** Αυτό σημαίνει ότι ΜΟΝΟ σε αυτό το αρχείο υπάρχει πρόβλημα.

Ανοίγοντας το αρχείο βλέπουμε το εξής

```
README.md (~/.myrossources/myTestRepo2LocalDir) - gedit
Open
<<<<<< HEAD
# myTestRepo2
=====
>>>>>> f8d94ec... My first commit!
```

Ας αναλυσουμε τη συνταξη του αρχιου

```
<<<<<<<< HEAD
```

```
# myTestRepo2
```

```
=====
```

Αυτο σημαινει πως κανονικα το αρχειο μας περιχει οτι βρισκεται αναμεσα στο

```
<<<<<<< και το =====
```

Απο το ===== μεχρι το >>>> ειναι αυτο που περιχει η αλλαγη του commit.

Στη συγκεκριμενη περιπτωση το περιεχομενο αναμεσα στο ===== και στο >>>> ειναι κενο.

αυτο σημαινει οτι το commit που κανουμε cherry-pick σβηνει οτι περιεχεται απο το <<<<

ως το ===== (το HEAD δηλαδη). Συνεπως η σωστη λυση εδω ειναι να σβησω τα παντα.

Αποθηκευω το αρχειο και αφου πλεον εχουμε κρατησει τις αλλαγες μας ξεκινουμε το git commit μας

```
git add -A
```

```
git commit
```

και παρατηρουμε οτι εχει ηδη συμπληρωσει το commit title και το description. Παταμε ctrl+χ για να βγουμε απο το nano. Πλεον μπορουμε να κανουμε push το commit.

```
git push origin master
```

Στη περιπτωση μας θα μας βγαλει προβλημα στο remote. Οποτε οπως και πριν

```
git remote add origin
```

Και ξανα git push origin master

```
~/myrossources/myTestRepo2LocalDir$ git add -A
~/myrossources/myTestRepo2LocalDir$ git commit
Recorded resolution for 'README.md'.
[master 7fe3622] My first commit!
Date: Fri Jun 9 16:22:05 2017 +0300
2 files changed, 1 insertion(+), 1 deletion(-)
create mode 100644 hello.txt
~/myrossources/myTestRepo2LocalDir$ git push origin master
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
~/myrossources/myTestRepo2LocalDir$ git remote add origin git@github.com:dkati/mytestrepo2.git
~/myrossources/myTestRepo2LocalDir$ git push origin master
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 313 bytes | 0 bytes/s, done.
Total 4 (delta 0), reused 1 (delta 0)
To git@github.com:dkati/mytestrepo2.git
5461521..7fe3622 master -> master
~/myrossources/myTestRepo2LocalDir$
```

Πραγματι παρατηρούμε ότι το commit έγινε επιτυχώς

USJ | <https://github.com/dkati/myTestRepo2/commits/master> | Αναζήτηση

This repository Search Pull requests Issues Marketplace Gist

dkati / myTestRepo2 Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

Branch: master

Commits on Jun 12, 2017

My first commit! dkati committed 3 days ago 7fe3622

Commits on Jun 9, 2017

Initial commit dkati committed 3 days ago 5461521

USJ | <https://github.com/dkati/myTestRepo2/commit/7fe36227387ab7e1cee0e61dff8ce7efedfa6cf6> | Αναζήτηση

This repository Search Pull requests Issues Marketplace Gist

dkati / myTestRepo2 Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

My first commit! Browse files

thats my first commit!YaY!

master

dkati committed 3 days ago 1 parent 5461521 commit 7fe36227387ab7e1cee0e61dff8ce7efedfa6cf6

Showing 2 changed files with 1 addition and 1 deletion. Unified Split

1 README.md View

... @@ -1,0 +0,0 @@

1 -# myTestRepo2

1 hello.txt View

... @@ -0,0 +1 @@

1 +hello

0 comments on commit 7fe3622

Αυτο που παρατηρουμε στα 2 commit ειναι οτι αλλες αλλαγες εγιναν στο commit του myTestRepo και αλλες εγιναν στο commit του myTestRepo2.

Αυτος ειναι και ο λογος που ειχαμε conflict κατα το cherry-pick.Ειναι σπανιες οι φορες που το cherrypick θα γινει χωρις conflict

Οταν ενα commit γινεται cherry-pick και ο συντακτης ειναι διαφορετικο ατομο απο αυτον που εκανε το cherry-pick τοτε το commit εμφανιζεται ετσι



sensitive_pn: Add Canadian sensitive numbers



Dmole committed with brinlyau 9 days ago

Ο πραγματικος συντακτης ειναι ο **brinlyau** ενω αυτος που εκανε το cherry-pick ειναι ο **Dmole**

ΕΠΑΝΕΓΓΡΑΦΗ ΤΗΣ ΙΣΤΟΡΙΑΣ ΤΩΝ COMMIT(ADVANCED USERS)

Οπως εχουμε αναφερει,δεν ειναι εφικτη η **ΔΙΑΓΡΑΦΗ** των commits.Παρολαυτα υπαρχει μια εντολη που επαναφερει **ΟΛΟΚΛΗΡΟ** το commit history σε μια παλαιότερη εκδοση σβηνοντας τα ενδιαμεσα commits.Η εντολη αυτη ειναι λιγο επικινδυνη καθως διαγραφει οτι commit υπαρχει

Πρακτικα δεν διαγραφει την υπαρξη τους απλα δεν φαινονται στο commit history.Επισης αναιρουνται απο τον τοπικο source code.Η διαδικασια αναιρεσης της διαγραφης ειναι δυσκολη και απαιτειται μεγαλη εξοικειωση με το github shell.

Η συνταξη της εντολης ειναι

Git reset --hard SHA_CODE (Διπλές παύλες)

Πχ Αν θελωνα «σβησω» το τελευταιο commit του myTestRepo2 ,τοτε αρκει να κανω reset στο προτελευταιο.

git reset --hard 5461521f50d472f7950f330608438a70634b435e

και μετα οφειλω να το κανω push

git push -f origin master

Το -f σημαινει «force» .Το github το κανει αυτο για ασφαλεια.αν δοκιμασουμε να κανουμε git push origin master δεν θα μας αφησει.

```
~/myrossources/myTestRepo2LocalDir$ git reset --hard 5461521f50d472f7950f330608438a70634b435e
HEAD is now at 5461521 Initial commit
~/myrossources/myTestRepo2LocalDir$ git push -f origin master
Warning: Permanently added the RSA host key for IP address '192.30.253.113' to the list of known hosts.
Total 0 (delta 0), reused 0 (delta 0)
To git@github.com:dkati/mytestrepo2.git
+ 7fe3622...5461521 master -> master (forced update)
~/myrossources/myTestRepo2LocalDir$
```


Αν το μετανιωσουμε μπορούμε να πάρουμε το σβησμένο commit (είναι το **7f3622**) κανοντας παλι reset

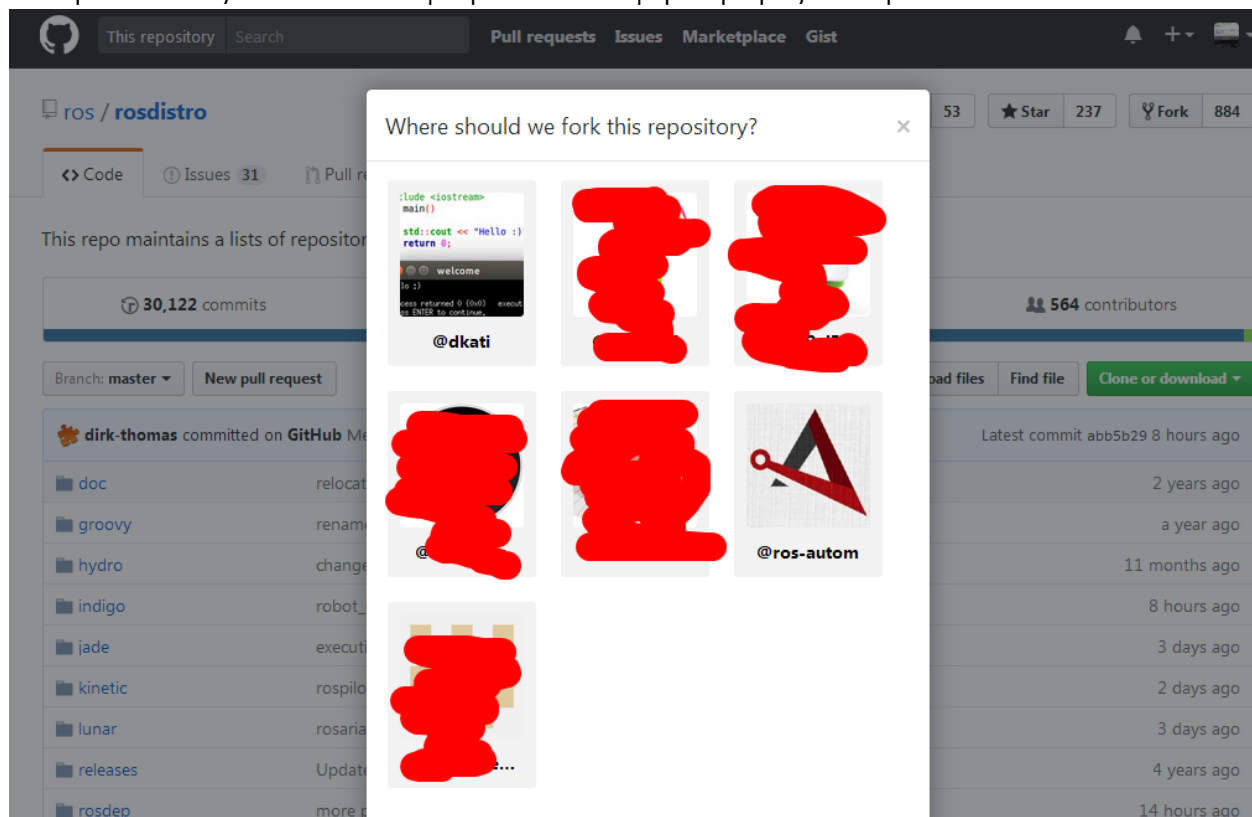
```
git reset --hard 7f3622
```

Συνεπώς καταλαβαίνουμε ότι το reset γράφει την ιστορία είτε προς τα πίσω είτε προς τα εμπρός.

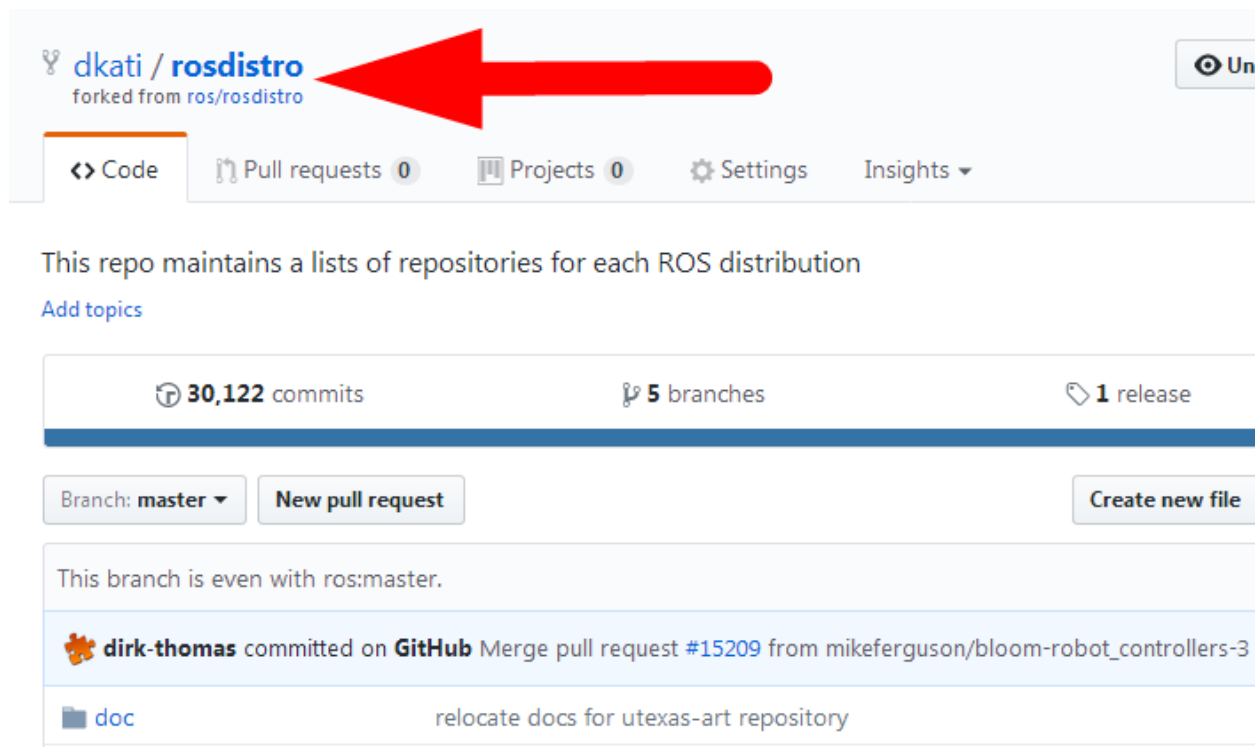
FORKING.Ο ΝΟΜΙΜΟΣ ΤΡΟΠΟΣ ΝΑ ΑΝΤΙΓΡΑΨΟΥΜΕ ΟΛΟΚΛΗΡΑ REPOSITORIES

Forking είναι η διαδικασία που αντιγράφουμε πλήρως ολόκληρο το repository καποιου. Είναι πολύ απλό και επιτυγχάνεται με 2 απλά click.

Πηγαίνουμε στο repository το οποίο θέλουμε να κάνουμε fork. Εστω το github.com/ros/rosdistro
Πατάμε πάνω δεξιά fork και επιλεγούμε σε ποιο λογαριασμό μας θέλουμε να παει.



Το repository εμφανίζεται στο προφίλ μας όπως παρακάτω.
φαινεται πολυ καθαρα οτι το repository ειναι εργο του ros/distro



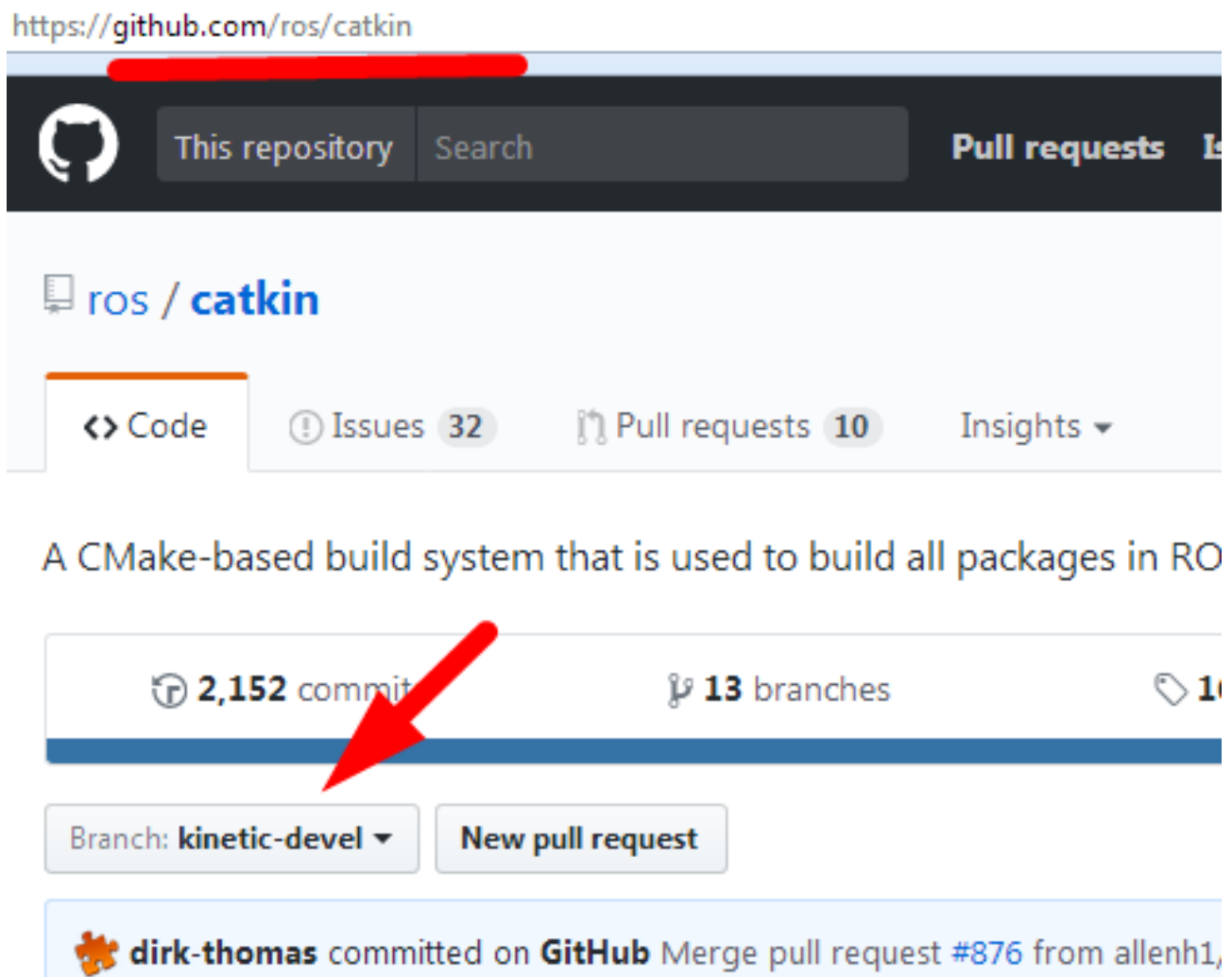
Απο εκει και περα χρησιμοποιω το project οπως ακριβως εργαστηκαμε με τα αλλα 2 repositories των παραδειγματων.

Παντοτε πρεπει να θυμομαστε οτι κατα το πρωτο push που θελουμε να κανουμε , χρειαζομαστε git remote add origin

ΓΡΑΨΙΜΟ ΟΛΟΚΛΗΡΗΣ ΙΣΤΟΡΙΑΣ ΑΠΟ ΑΛΛΟ REPOSITORY

Εστω οτι εχω ενα repository με branch master και θελω να φτιαξω ενα ολοκληρο branch το οποιο θα ειναι πρακτικα,το branch ενος τριτου repository.

Εστω λοιπον οτι θελω στο mytestrepo να φτιαξω ενα branch που θα περιεχει το kinetic-devel branch απο το ros/catkin repository



Προφανως δεν γινεται να κανω fork διοτι δεν θελω ΟΛΑ τα branch.Συνεπως πρεπει να δουλεψουμε ως εξης

- Τραβαω ολοκληρο το repository.
- Μπαινω στο kinetic-devel branch
- Φτιαχνω τοπικα ενα branch το οποιο προερχεται απο το kinetic-devel (βλ.την εικονα με τα βελακια και τα κυκλακια)
- Το κανω push

Συνεπώς μπαίνουμε στο myTestRepoLocalDir

```
git clone https://github.com/ros/catkin kinetic-devel
cd kinetic-devel
git branch
```

Παρατηρούμε ότι είμαστε στο branch που έχουμε κατεβάσει. Απο το σημείο αυτό φτιάχνουμε το δικό μας branch.

```
git branch mykinetic
git branch
```

Πλέον βλέπω ότι έχω 2 branches και ξέρω ότι το mykinetic έχει προερθεί από το kinetic-devel. Με πράσινο χρώμα είναι το τοπικό branch στο οποίο βρισκόμαστε. Αλλάζουμε λοιπόν το branch

```
git checkout mykinetic
```

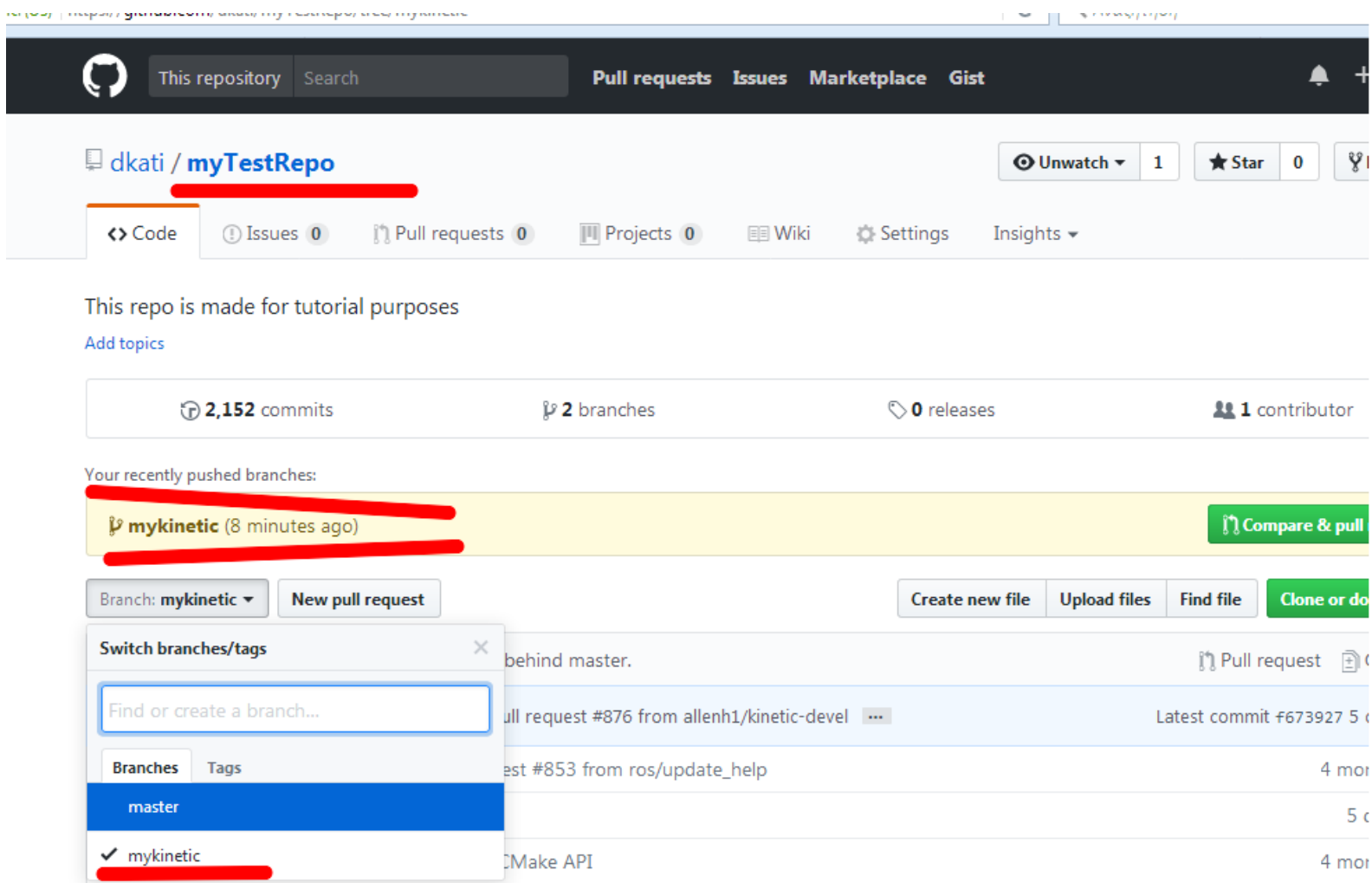
Αφού είμαι στο branch, κάνω push ότι υπάρχει στο branch (και sourcecode και commit history). Αν γράψουμε git push origin mykinetic θα μας πει ότι δεν έχουμε δικαιώματα να κάνουμε push το οποίο είναι προφανές καθώς δεν ανήκουμε στην ομάδα του ros. Αρα πρέπει να φτιάξουμε νέο origin

```
git remote add myorigin
git push -f myorigin mykinetic
```

Χρησιμοποιώ -f γιατί αναγκάζω το github να μου φτιάξει στη σελίδα νέο branch

```
~/myrossources/myTestRepoLocalDir$ git clone https://github.com/ros/catkin kinetic-devel
Cloning into 'kinetic-devel'...
remote: Counting objects: 12110, done.
remote: Total 12110 (delta 0), reused 0 (delta 0), pack-reused 12109
Receiving objects: 100% (12110/12110), 3.34 MiB | 1.26 MiB/s, done.
Resolving deltas: 100% (6467/6467), done.
Checking connectivity... done.
~/myrossources/myTestRepoLocalDir$ cd kinetic-devel
~/myrossources/myTestRepoLocalDir/kinetic-devel$ git branch
* kinetic-devel
~/myrossources/myTestRepoLocalDir/kinetic-devel$ git branch mykinetic
~/myrossources/myTestRepoLocalDir/kinetic-devel$ git branch
* kinetic-devel
  mykinetic
~/myrossources/myTestRepoLocalDir/kinetic-devel$ git checkout mykinetic
Switched to branch 'mykinetic'
~/myrossources/myTestRepoLocalDir/kinetic-devel$ git push -f origin mykinetic
Username for 'https://github.com': dkati
Password for 'https://dkati@github.com':
remote: Permission to ros/catkin.git denied to dkati.
fatal: unable to access 'https://github.com/ros/catkin/': The requested URL returned error: 403
~/myrossources/myTestRepoLocalDir/kinetic-devel$ git remote add myorigin git@github.com:dkati/mytestrepo.git
~/myrossources/myTestRepoLocalDir/kinetic-devel$ git push -f myorigin mykinetic
Counting objects: 11112, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4919/4919), done.
Writing objects: 100% (11112/11112), 3.15 MiB | 1.23 MiB/s, done.
Total 11112 (delta 5839), reused 11108 (delta 5836)
remote: Resolving deltas: 100% (5839/5839), done.
To git@github.com:dkati/mytestrepo.git
 * [new branch]    mykinetic -> mykinetic
~/myrossources/myTestRepoLocalDir/kinetic-devel$
```

Εν τέλει στο repository μας παρατηρούμε το νέο branchτο οποίο είναι ΑΚΡΙΒΩΣ ίδιο με το πραγματικό branch του ros



Αφου τελειωσαμε με το github,ας κανουμε ενα καθαρισμο τον κωδικα μας

```
cd ../../  
rm -rf myTestRepoLocalDir  
repo sync myTestRepoLocalDir --force-sync
```

--force-sync καλο ειναι να χρησιμοποιουμε σε καθε reposync για να το αναγκασουμε να «τραβηξει» καθε νεα αλλαγη και να σβησει τυχον ξεχασμενα λαθη στο τοπικο source

Πολυ σημαντικό είναι να παρατηρήσουμε ότι ακόμη και το commit history είναι ΑΚΡΙΒΩΣ ίδιο με το αυθεντικό repository του ros

https://github.com/dkati/myTestRepo/commits/mykinetic

This repository Search Pull requests

dkati / **myTestRepo**

<> Code Issues 0 Pull requests 0 Projects

Branch: mykinetic

Commits on Jun 7, 2017

- Merge pull request #876 from allenh1/kinetic-d
dirk-thomas committed on GitHub 5 days ago
- Fixed a typo.
allenh1 committed 5 days ago

Commits on Mar 29, 2017

- Merge pull request #862 from ros/support_exte
dirk-thomas committed on GitHub on Mar 29
- use environment variable to extend environmen
dirk-thomas committed on Mar 29

Commits on Mar 10, 2017

- Merge pull request #861 from ros/report_proble
dirk-thomas committed on GitHub on Mar 10

Commits · ros/catkin

GitHub, Inc. (US) https://github.com/ros/catkin/commits/kinetic-devel

This repository Search Pull requests Issues Marketplace

ros / **catkin**

<> Code Issues 32 Pull requests 10 Insights

Branch: kinetic-devel

Commits on Jun 7, 2017

- Merge pull request #876 from allenh1/kinetic-devel
dirk-thomas committed on GitHub 5 days ago ✓
- Fixed a typo.
allenh1 committed 5 days ago ✓

Commits on Mar 29, 2017

- Merge pull request #862 from ros/support_extend_in_plain_shell
dirk-thomas committed on GitHub on Mar 29 ✓
- use environment variable to extend environment in plain shell
dirk-thomas committed on Mar 29 ✓

Commits on Mar 10, 2017

- Merge pull request #861 from ros/report_problems_msg
dirk-thomas committed on GitHub on Mar 10

- `git branch`
Εμφάνιση των διαθεσιμων τοπικων branches.Επισης εμφανιζει το branch στο οποιο ειμαστε τωρα(εμφανιζεται με πρασινα γραμματα).
- `git branch <branch name>`
Δημιουργια ενος νεου branch ,απο το branch που ειμαστε ηδη
- `git checkout <branch name>`
Μεταβαση σε αλλο branch
- `git checkout -b <branch name>`
Δημιουργια ενος νεου branch και μεταβαση σε αυτο
- `git branch -D <branch name>`
Διαγραφη ενος branch.ΔΕΝ ΜΠΟΡΟΥΜΕ ΝΑ ΔΙΑΓΡΑΨΟΥΜΕ ΤΟ BRANCH ΣΤΟ ΟΠΟΙΟ ΕΙΜΑΣΤΕ.
- `git clone <github link απο repository> <directory>`
Κατεβασμα του repository (default branch) και αποθηκευση σε ενα νεο φακελο με ονομα το ονομα του directory.Ο φακελος βρισκεται στο ιδιο directory που ειμαστε με το terminal
- `git clone -b <branch name> <github link απο repository> <directory>`
Κατέβασμα του repository (Το branch που εχουμε δωσει) και αποθηκευση σε ενα νεο φακελο με ονομα το ονομα του directory.Ο φακελος βρισκεται στο ιδιο directory που ειμαστε με το terminal
- `git fetch <github link απο repository> <branch name>`
Κατεβασμα του commit history του repository.ΠΡΕΠΕΙ ΝΑ ΤΡΕΞΕΙ ΜΕΣΑ ΣΤΟ DIRECTORY ΣΤΟ ΟΠΟΙΟ ΘΑ ΚΑΝΩ ΤΟ cherry-pick
- `repo init -u git://<link απο manifest>.git -b <branch name>`
Αρχικοποιηση του manifest
πχ `repo init -u git://github.com/dkati/myproject-manifest.git -b master`
- `git cherry-pick<SHA>`
Νομιμη αντιγραφη ενος commit με συγκεκριμενο SHA
- `git reset --hard <SHA>`
Επαναφορα του commit history στο συγκεκριμενο commit
- `git revert <SHA>`
Αναστροφη του συγκεκριμενου commit
- `git commit`
Δημιουργια commit
- `git add -A`
Προσθηκη των αλλαγων μου στο commit
- `git push <remote name><branch name>`
Ανεβασμα του commit ,μεσω του remote,στο branch name
- `git remote add <remote name> git@github.com:<repo name>.git`
Δημιουργια remote
- `git remote remove <remote name>`
Σβησιμο remote

- Repo sync ή repo sync --force-sync
Συγχρονισμός τοπικού source code με αυτο στο github.
--force-sync αν θελω να αναγκασω το github να σβησει τις τοπικες αλλαγες μου

Ολες οι εντολες πρεπει να εκτελουνται μεσα στο root directory του καθε repository.Εκει δηλαδη οπου υπαρχει φακελος .git

***=

Οταν το github μας αναφερει Recorded preimage σημαινει οτι εχει καταγραφει το λαθος ΚΑΙ ΤΗΝ ΕΠΙΛΥΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ του χρηστη .Αυτο το κανει για να λυσηι αυτοματα το ιδιο προβλημα σε καποιο μελλοντικο cherry-pick.Πολλες φορες ομως δεν θελουμε να θυμαται το πως το λυσαμε γιατι ενδεχομενωσ η λυση να μην ειναι ιδια.Ετσι λοιπον χρησιμοποιω την εντολη

git rerere forget *

Με την εντολη αυτη λοιπον αναγκαζω το github να «ξεχασει» οποιαδηποτε πιθανη λυση ενος conflict.