



Εγχειρίδιοχρήσης GitHub   
μέσω Linux Terminals  
ROS-autom 2017 ®

Github

# προετοιμασια συστηματοσ

Επαληθευουμε οτι εχουμε τα απαραιτητα πακετα

**sudo apt-get install git  
sudo apt-get install repo**

Το πιο πιθανο ειναι να μην υπαρχει μονο το repoκαι να απαιτει εγκατασταση.Στη συνεχεια πρεπει να συνδεθει το githubclientτου υπολογιστη μας,με το githubστη σελιδα,ωστε να μπορω στη συνεχεια να κανω commits,pulls,pushesκτλπ.

Τρεχουμε

**git config –global user.email** [**dkatikaridis@gmail.com**](mailto:dkatikaridis@gmail.com)**(((διπλέςπαυλες)  
git config –global user.name mygithubname**

Tο mailκαι ο κωδικος πρεπεινα ειναι σε πληρη αντιστοιχεια με αυτα που βλεπουμε στη σελιδα.  
τα δικα μου ειναι [dkatikaridis@gmail.com](mailto:dkatikaridis@gmail.com)με usernamedkati

Στη συνεχεια πρεπει να πιστοποιησουμε στο github ,τον υπολογιστηπου θα κανει push  
απο το λογαριασμο μας.

**ssh-keygen -t rsa -b 4096 -C** [**dkatikaridis@gmail.com**](mailto:dkatikaridis@gmail.com)

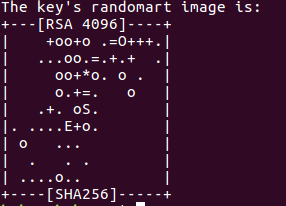
Καιθαπαρουμεωςεπιστροφη

**Generating public/private rsa key pair.**

**Enter file in which to save the key  
παταμε enter**

**Enter passphrase   
παλι enter**

**Enter passphrase again**

**Παλι enter  
Τελικα θα μα δωσει το RSA 4096**  


Στη συνεχεια πρεπει αυτο το sshkeyνα γραφτεί στον ssh-agent.

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

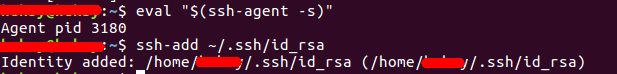
Θα παρουμε μια εξοδο σαν αυτη : **AgentpidXXXXX**

Αποθηκευουμε το sshσε ενα τοπικο αρχειο

**ssh-add ~/.ssh/id\_rsa**

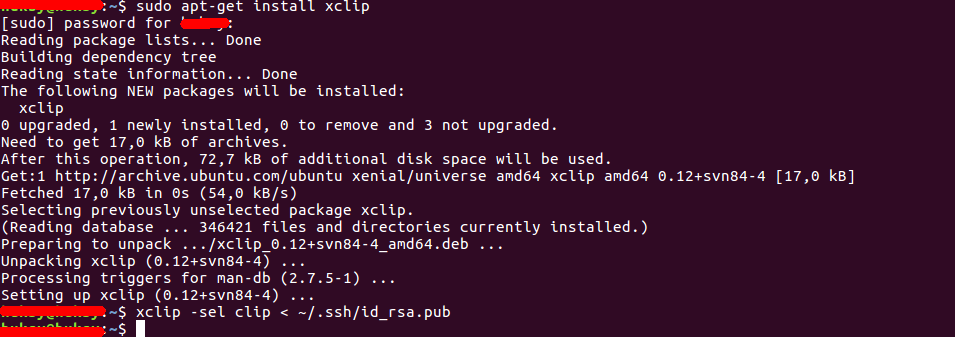
Με εξοδο

**Identity added : /home/username/.ssh/id\_rsa ( home/username/.ssh/id\_rsa )**

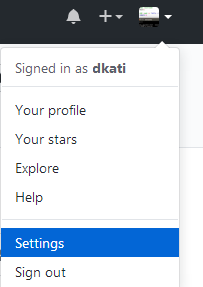


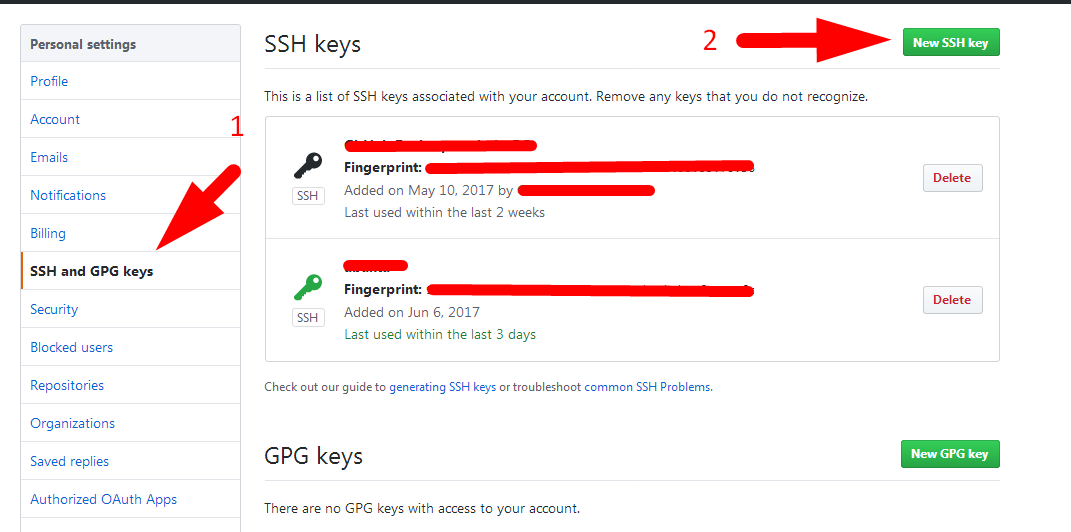
Στη συνεχεια πρεπει να αντιγραψουμε το SSHστο clipboard.Ετσι κατεβαζουμε το εργαλειο xclip  
και αντιγραφουμε το SSHμεσω αυτου,στο clipboard

**sudo apt-get install xclip && xclip -sel clip < ~/.ssh/id\_rsa.pub**

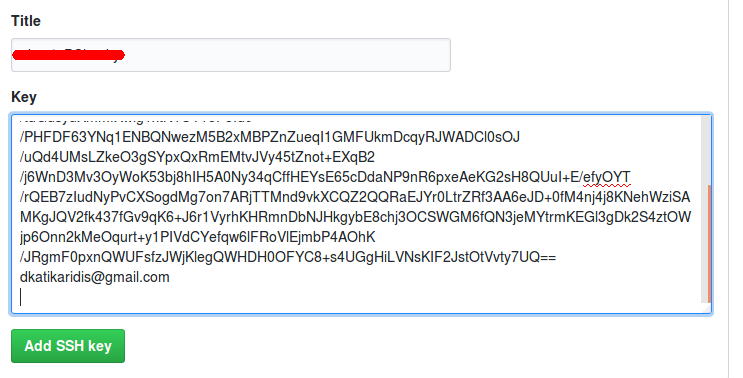


Στη συνεχεια αυτο το sshπρεπει να προσθεθει στο λογαριασμο μας στο github.Ανοιγουμε τις ρυθμισεις του λογαριασμου μας

  
Επιλεγουμε απο τα μενου , SSHandGPGkeysκαι παταμε στο δεξια κουμπι «NewSSHkey».  
Bαζουμε στο Titleεναν τιτλοκαι στο πλαισιο keyαπλα παταμε δεξι κλικ/επικολληση



To τελικο αποτελεσμα θα πρεπει να ειναι ετσι



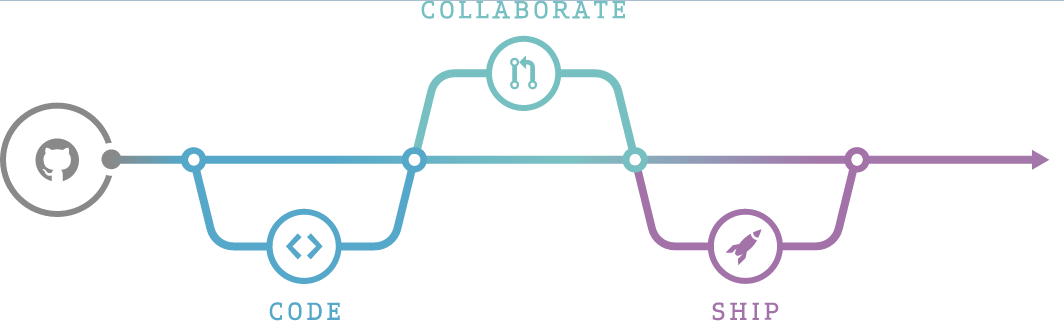
Και παταμε AddSSHkey.

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

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

Github

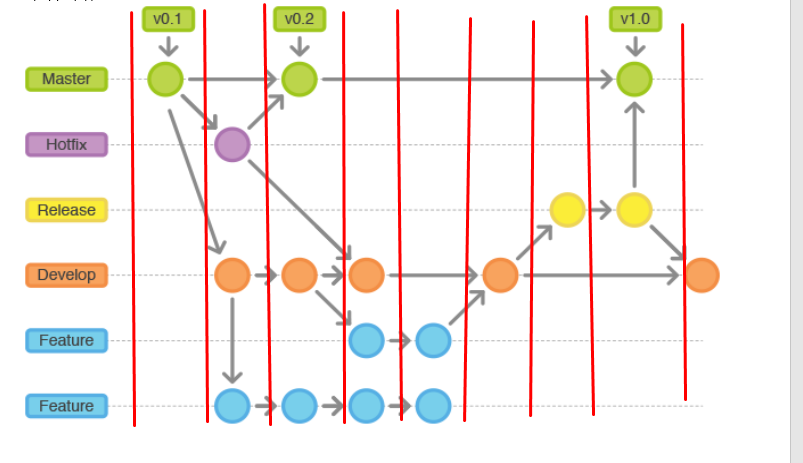
# TΡΟΠΟΣ ΛΕΙΤΟΥΡΓΙΑΣ ΚΑΙ ΦΙΛΟΣΟΦΙΑ



Τη λειτουργια του githubμπορουμε να τη φανταστουμε ως ενα δεντρο (tree) με κλαδια (branches) και κυριο κορμο(masterbranch),πανω σε ενα χρονοδιαγραμμα.Για την ακριβεια ως ενα δεντρο με κερασια(Θα αναλυθει παρακατω).Παρακατω θα μελετησουμε ενα flowchartμιαςτυχαιας εφαρμογης

Στην εικονα παρατηρουμε το εξης.

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

Το κυριο πλαισιο ειναι το master.Είναι το πρωτο branchπου δημιουργειται και η κυρια ροη του προγραμματος.Τα υπολοιπα πλαισια ειναι τα δευτερευοντα branches.  
Οι διακεκομμενες γραμμες ειναι οι οριζοντιες γραμμες που μας δειχνουν την χρονικη εξελιξη των πραγματων απο αριστερα προς τα δεξια.   
Τα βελακια ροης δειχνουν τα βηματα που κανει η ροη του github το οποιο θα αναλυσουμε παρακατω.  
Tα κυκλακια αντιπροσωπευουν μια αλλαγη(ενα commit) στον κωδικα.  
Πολυ σημαντικο ειναι να παρατηρησουμε πως τα κυκλακια βρισκονται σε καθετη αντιστοιχια.  
  


Ας το αναλυσουμε αυτο.Στο πρωτο καθετο πλαισιο υπαρχει ΜΟΝΟ το κυκλακι του masterbranch.  
Αυτο σημαινει οτι ειναι το ΠΡΩΤΟ-initialreleaseτου sourcecodeμας στο github.Mπορουμε να φανταστουμε το masterbranchως το «επισημο» sourcecodeπου θα θελαμε καποιος να δει..Συνηθως ειναι η stableεκδοση του τρεχοντα κωδικα.

Στη συνεχεια παρατηρουμε 3 βελακια.  
1 προς το hotfix.1 προς το develop.1 προς το feature.  
  
Αυτο σημαινει οτι απο το masterφτιαξαμε αλλα 3 branches.Βλεπουμε πως τα κυκλακια αυτα δεν ειναι στην ιδια καθετο με το master.αυτο σημαινει πως εχει γινει καποιο commit-καποια αλλαγη στον κωδικα.  
Συνεπως βλεπουμε οτι ο προγραμματιστης του κωδικα εφτιαξε 3 ακομα branchγια να μπορει να προσθεσει μια αλλαγη.  
-Και γιατι δεν βαζει την αλλαγη κατευθειαν στο masterbranch?  
-Για να μπορει να την ελεγξει.Αν δουλευει σωστα τοτε την προσθετει στο masterbranchπου θα δουμε παρακατω  
Στη τριτη καθετη γραμμη βλεπουμε το hotfixνα μπαινει στο master.Αυτο σημαινει πως το hotfixηταν πιθανον ενα bugfix ,οποτε ο προγραμματιστης το προσαρμοσε στο κυριo branch.Επισης ο προγραμματιστης ανεβαζει την εκδοση του προγραμματος σε v0.2

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

Καθε κυκλακι ειναι και ενα commit(μια αλλαγη) στον κωδικα.Αυτο σημαινει 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 ,σημαινει πως απο εκεινο το σημειο(commit) ,φτιαχνω νεο branch.Θα τα αναλυσουμε ολα αυτα και παρακατω

**Παρατηρησεις**

* Το githubκραταει ιστορικο των commits .Αν κατι γραφτει στο ιστορικο ,ΔΕΝ ΔΙΑΓΡΑΦΕΤΑΙ.
* Το ιστορικο του githubμπορει να αναιρεθει (Να γινει reset) ;ή να γινει Revert
  + Revertσημαινει να γυρισω τον κωδικα πως ηταν πριν.  
    Αν πχ,εσβησα ενα declarationμιας μεταβλητης,τοτε με το revertτη ξαναδηλωνει
* Ολοκληρο το sourcecodeμε τα ολα τα branchesονομαζεται repository (repo)
* Το οτι κανω pushμια αλλαγη/ενα commitδεν σημαινει πως αλλαζω branch.Ο λογος που αλλαζω branchειναι για να μπορω να κανω δοκιμες στον κωδικα χωρις να επιρρεαζω τον βασικο κωδικα που θεωρω stable
* Μεσα σε ενα repositoryμπορουμε να προσθεσουμε contributorsκαι 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 καιgitεντολες

Το githubαρχικα δημιουργηθηκε με μοναδικο τροπο χρησης ,τα linuxterminals.Αργοτερα δημιουργηθηκε και η desktopεφαρμογη για Windows.Εδω θα αναλυθει η χρηση μεσω linuxterminal.

Η εξοικειωση με το githubμεσω terminalθα βοηθησει και αυτον που θα θελει να χρησιμοποιησει την windowsdesktopεφαρμογη.**Oχι ομως το αντιστροφο**(Η εφαρμογη για windows περιεχει επισης terminal ,για advanced χρηστες)

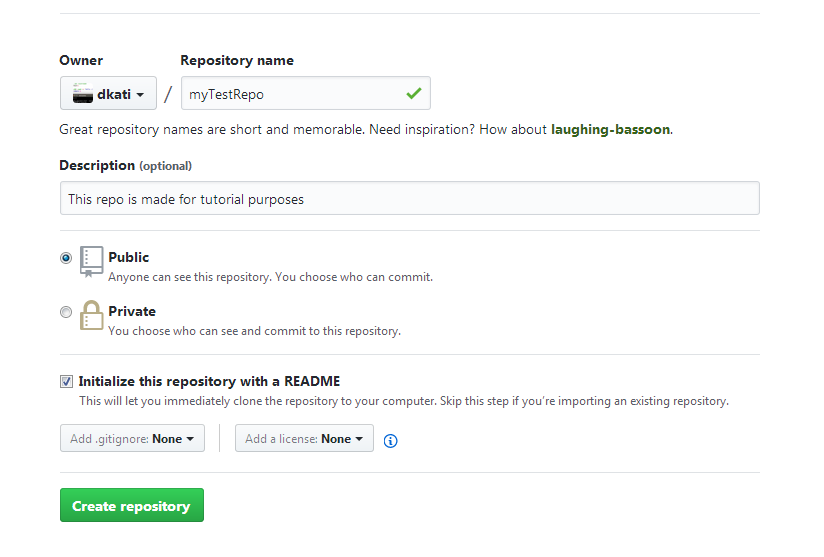
Οι τροποι που μπορω να ανεβασω/κατεβασω τον κωδικα μου στο/απο το githubειναι δυο.

* Ανεβασμα/Κατεβασμα τον κωδικα σε συγκεκριμενο branch ,στο githubμου, και επεξεργασια.Το μειονεκτημα εδω ειναι πως ορισμενες φορες πρεπει να κατεβαζουμε ΟΛΟΚΛΗΡΟ τον κωδικα απο το githubκαι οχι μονο τα τελευταια commitsπου καποιος αλλος contributor εκανε στο project
* Δημιουργια manifestπου εχει κατεβασμενα ολα τα branchσε .tmpφακελο και ειναι ΠΑΝΤΑ προσβασιμα χωρις πιθανοτητα διαγραφης.Το μειονεκτημα εδω ειναι πως ο τροπος αυτος ειναι λιγο πιο περιπλοκος

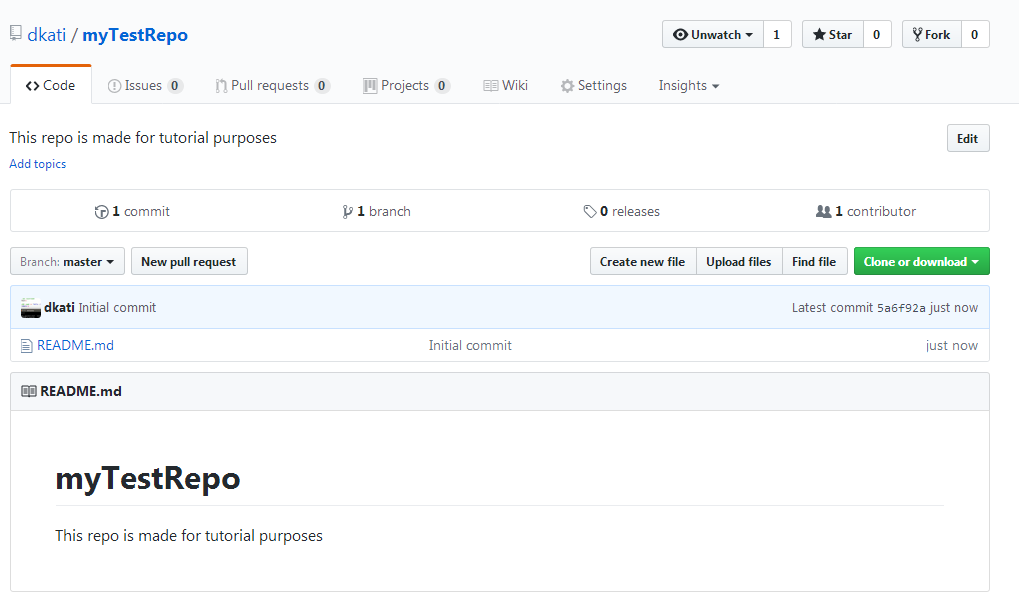
Παρακατω θα αναλυθει ο **δευτερος** τροπος τον οποιο θεωρουμε πιο ασφαλες.Παρολαυτα θα αναφερθουμε και στον πρωτο τροπο ο οποιος μας βοηθαει στο να παρουμε ολοκληρα repositoryαπο αλλους

# Ξεκινωντας νεο repositoryγια να ανεβασουμε τον κωδικα ενοσ project που εχουμε γραψει

Μεταφερομαστε στο προφιλ μας στο github.com.Αφου εχουμε ηδη κανει login , επιλεγουμε το **+** απο πανω δεξια και επιλεγουμε Newrepository.Εκει μπορουμε να συμπληρωσουμε τα στοιχεια του αρχικου repository.Αρχικα θα ειναι αδειο και μετα θα κανουμε προσθηκη του κωδικα μας ως initialrelease.Γραφουμε το ονομα του repository ,μια περιγραφη εαν θελουμε,Δημοσιο repositoryκαι επιλεγουμε και το InitializethisrepositorywithaREADME για να δημιουργησει το πρωτο μας αρχειο.Αργοτερα θα προσθεσουμε και το license.

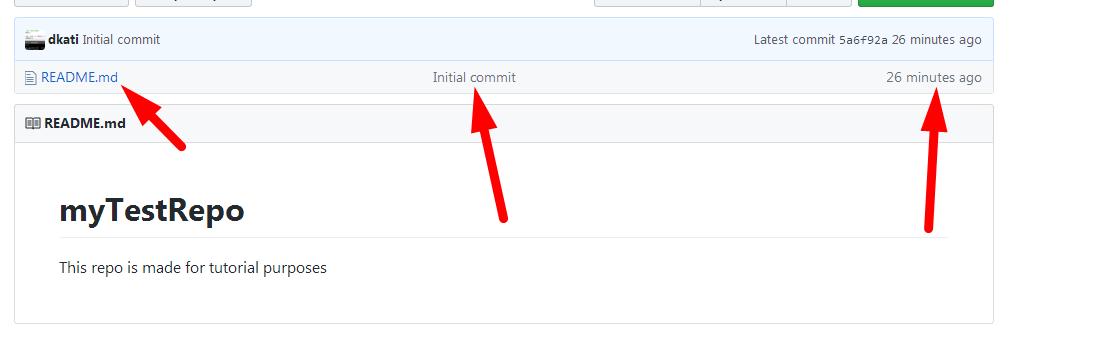


Αφου πατησουμε Createrepositoryθα δουμε την εικονα αυτη



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 contributorsνα τα δουν και να απαντησουν/λυσουν τα ζητηματα
  + Pullrequests : Στη καρτελα αυτη ειναι μαζεμενες καποιες προτασεις που καποιος τριτος χρηστης κανει,σχετικα με τον κωδικα.Οι contributorsβλεπουν τις αλλαγες και αν θελουν με το πατημα ενος κουμπιου επιτρεπουν το τριτο χρηστη να προσθεσει τα commitτου στο repositoryμας,χωρις να ειναι μελος αυτου
  + Wiki : Το παραδοσιακο wikiοπου οι contributorsδινουν καποιες οδηγιες σχετικα με το sourcecode
  + Settings : Ρυθμισειςτουrepository(Οχιτουsource code)
* Παρακατω ειναι το descriptionπου εχουμε βαλει στο repository
* Στη συνεχεια υπαρχει αλλο ενα μενου
  + 1 Commit : Πατωντας πανω στο μενου αυτο μας εμφανιζει ολα τα commitsπου εχουν γινει με χρονολογικη σειρα.Λεπτομερειες για τα commitsθα αναφερθουν αργοτερα
  + 1 branch : εμφανιζει τα branches
  + 0 releases : αφορα τα releaseπου κανουν οι contributors
  + 1 contributor : εμφανιζει ολους οσους συνεισφερει στον κωδικα ειτε ειναι μελη του repositoryειτε χρησιμοποιησαμε τον κωδικα του
* Παρακατω βλεπουμε ενα κουμπι-μενου που λεει branch: master.Απο εδω μπορω να αλλαζω branchesκαι να βλεπω τον αντιστοιχο κωδικα και τα αντιστοιχα commits.
* Newpullrequest: Εαν θελω ως τριτος χρηστης να προσθεσω κωδικα
* Createnewfile/Uploadfiles : Χειροκινητη δημιουργια/δημοσιευση αρχειου **(Δεν συνισταται)**
* Clone/Download : Κατεβασμα του sourcecode σε zip μορφη (Δεν συνισταται!Ενα λογος που δεν συνισταται ειναι οτι στα linux περιβαλλοντα ,το zip αρχειο μπορει να διαγραψει τυχον symlinks κατα το extract
* **dkati** Initialcommit : Αναφερεται το τελευταιο commitπου εχει γινει.Το githubαυτοματα κανει ενα commitοταν δημιουργουμε το repositoryκαι προσθετουμε readme.Η μορφη του τιτλου του commitειναι   
  <githubusername><Τιτλος commit>
* Ολα τα υπολοιπα απο κατω ειναι τα αρχεια του συγκεκριμενου directoryμαζι με τις λεπτομερειες



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

Το μεσαιο βελακι δειχνει το τελευταιο commitπου εχει γινει **και εχει επιρρεασει το συγκεκριμενο αρχειο**

Το δεξια βελακι δειχνει την ωρα που εχει περασει απο το τελευταιο commitπου επιρρεασε το αντιστοιχο αρχειο.

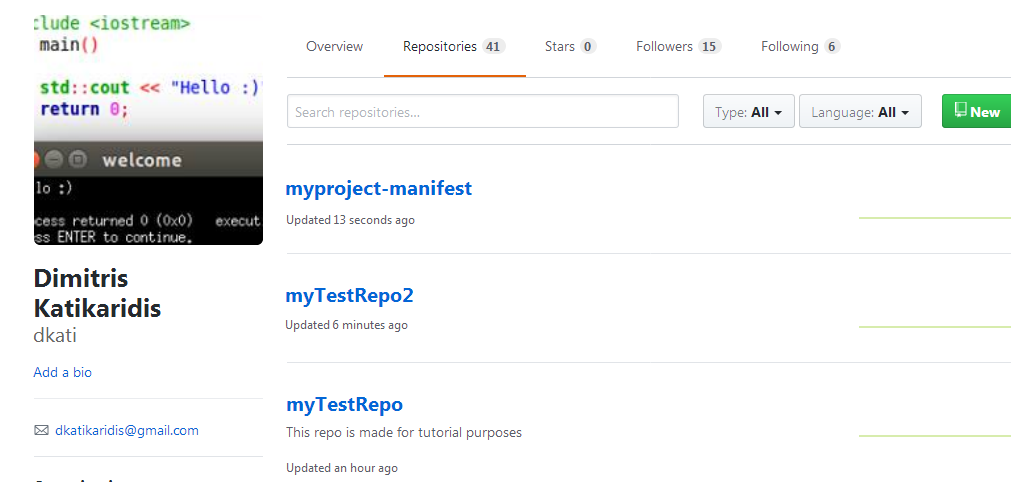
Γενικοτερα οι πληροφοριες αυτες μας βοηθουν να εχουμε μια ταξη μεσα σε τεραστιου μεγεθους κωδικες

Δεδομενου λοιπον οτι καταλαβαμε πως ειναι η δομη του repositoryπαμε να φτιαξουμε **ΑΛΛΟ ΕΝΑ repository με ονομα myTestRepo2**το οποιο θα ειναι το 2ο εργαλειο μας και στη συνεχεια θα εξηγησουμε και θα φτιαξουμε το manifest που τα συνδεει ολα αυτα μαζι.

Manifestειναι ενα ειδικο repositoryπου περιεχει ενα ή και παραπανω αρχεια xml τα οποια περιεχουντα githublinksαπο τα projectsπου χρειαζομαστε για να στησουμε ενα ολοκληρο sourcecode.  
Εχουμε παρατηρησει οτι πολλα sourcecodeαπο επαγγελματικα προγραμματα ειναι χωρισμενα σε διαφορα μερη ,που καθε μερος ειναι ενα εργαλειο.Λογου χαρειν,στο ROSεχουμε διαφορα εργαλεια (Gazebo,ROScore,RVIZ) τα οποια θελουμε να δουλεουν ολα μαζι.  
Ετσι λοιπον εμεις θα φτιαξουμε εστω 2 repositories για καθε ενα εργαλειο που θελουμε να κανουμε τις δικες μας/customαλλαγες.  
Η δουλεια του manifestειναι να οριζει στο gitservice ,ποια sourcesπρεπει να κατεβουν.  
Αντι λοιπον να κατεβαζουμε ενα-ενα τα projectμας ,τα τοποθετουμε σε ενα directoryκαι το χωριζουμε σε κομματια μεσω του manifest για καλυτερη οργανωση.

# Δημιουργια ενοσ manifest

Δημιουργουμεενανεοδημοσιοrepositoryμεονομαmyproject-manifest ,μεreadme.  
Πριν προχωρησουμε ας δουμε λιγο την αρχικη οθονη του προφιλ μας.



Παρατηρουμε οτι πλεον εχουμε 3 repositories.Το ενα ειναι το myTestRepoπου ενδεχομενως να ειναι το ενα απο τα εργαλεια που sourcecodeμου ,το δευτερο ειναι το myTestRepo2 το οποιο ειναι καποιο 2 εργαλειο μου και το αλλο ειναι το manifest

Παταμεπανωστοmyproject-manifestκαι επιλεγουμεαποδεξια «Createnewfile»   
Στηνεπιλογηονοματοςπληκτρολογουμε «default.xml» καιωςπεριεχομενοβαζουμε

<?xmlversion=**"1.0"**encoding=**"UTF-8"**?>

<manifest>

<remotename=**"github"**

fetch=**"https://github.com/"**/>

<projectpath=**"myTestRepoLocalDir"**name=**"dkati/myTestRepo"**remote=**"github"**revision =**"master"**/>

<projectpath=**"myTestRepo2LocalDir"**name=**"dkati/myTestRepo2"**remote=**"github"**revision =**"master"**/>

</manifest>

Ας εξηγησουμε τον κωδικα

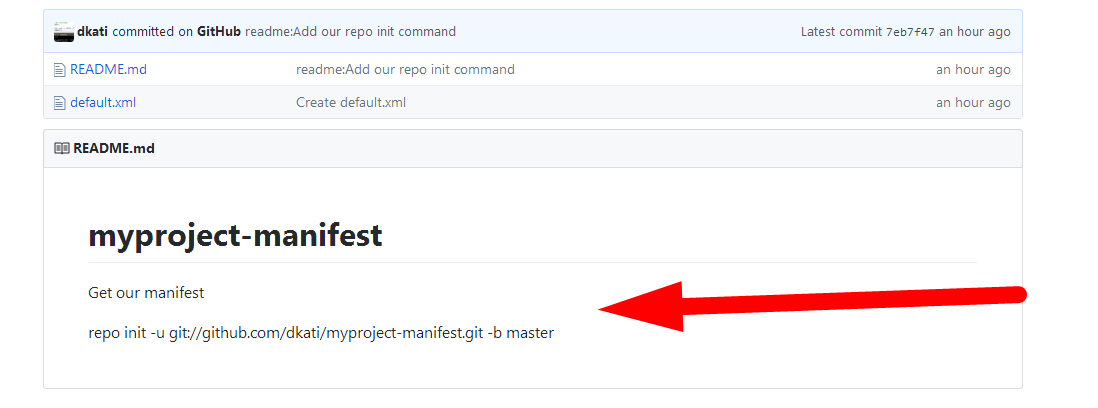
Η πρωτη σειρα αφορα την μορφη του XML.  
remoteειναι το μελος του xmlπου οριζει απο που θα τραβαει το gitservice,ολα τα sources.  
το μελος pathτου project,ειναι το τοπικο directoryπου θελουμε να παει το κατεβασμενο sourcecode  
Tonameειναι το githubnameτου repository  
Revisionειναι το branch το οποιο θα εμφανιζεται στον τοπικο φακελο μας και το branchτο οποιο θα κατεβαζουμε/κανουμε push.

Κατω απο την επικεφαλιδα Commitnewfile οριζουμε το Committitle.  
Εδω οριζουμε τον τιτλο του commit που θα φαινεται στα commits.Γενικοτερα πρεπει να προσεχουμε τα commit title που κανουμε καθως ειναι αυτα που καποιος τριτος βλεπει.Οποτε η εικονα που βγαινει προς τα εξω πρεπει να ειναι προσεγμενη.Ενα τυπικο προτυπο committitle ειναι το εξης:  
  
<dir></subdir> : <Τι εχω αλλαξει>  
Πχ αν εκανα αλλαγες μεσα σε μια class σε ενα αρχειο στο directorymysource/src/libs/mylib.cpp  
Τοτε στο committitle μπορω να γραψω ->src/lib: Donotexpose \_varfrommylib  
ή κατι παρομοιο.Γενικοτερα προσπαθουμε να κανουμε ευστοχα committitles χωρις μεγαλη εκταση  
Στο description βαζουμε την περιγραφη του commit,αν θελουμε να εξηγησουμε κατι,και παταμε commitnewfile  
Mετα απο τη δημιουργια του αρχειου μπαινουμε στο αρχειο readme.md και επιλεγουμε το δεξια μολυβι ωστε να το επεξεργαστουμε.Μεσα στο αρχειο θα γραψουμε την κυρια εντολη που μας κατεβαζει το manifest στον υπολογιστη και το ρυθμιζει.Η εντολη ειναι η   
  
repoinit -ugit://github.com/dkati/myproject-manifest.git -bmaster

Στη συνεχεια πρεπει να συνταξουμε ενα committitle.  
Ενας αποδεκτος τιτλος θα μπορουσε να ειναι ->**readme:Addourrepoinitcommand**

Aφου κανουμε το commitεπιστρεφουμε στο myproject-manifest

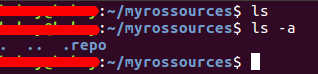
Παρατηρουμε οτι το επεξεργασμενο readmeφαινεται μπροστα στο repository.**Αυτο ισχυει μονο για τα αρχεια readme**.Δεν σημαινει πως οποιαδηποτε αλλαγη σε αρχειο θα εμφανιζεται μπροστα.



# αρχικοποιηση του manifestκαι των sourcecodes

Απο τη στιγμη που ετοιμασαμε τα repositoriesκαι το manifest,μπορουμε πλεον να ξεκινησουμε τη διαδικασια της αποθηκευσης των repositoriesαυτων,στον υπολογιστη μας.Εκτελουμε   
**cd  
mkdirmyrossources&&cdmyrossources**

Aν μας βγαλει μηνυμα σχετικα με τα χρωματα του λογαριασμου παταμε ‘Y’ και προχωραμε.Με μια πρωτη ματια μεσα στο φακελο δεν φαινεται να υπαρχουν αρχεια.Παρολαυτα με την εντολη   
ls -aπαρατηρουμε οτι υπαρχει ενα νεος φακελος .repo

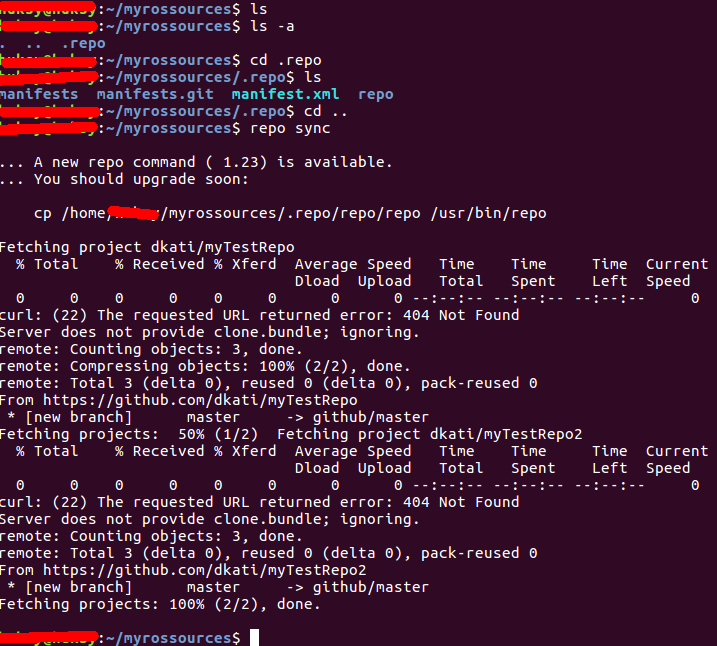


Στοφακελοαυτουπαρχειτοsymlinkmanifest.xmlτοοποιοδειχνειστο**.repo/manifests/default.xml**  
τοοποιοειναιτοmanifestπου γραψαμε πριν.

Ο φακελος repoεχει μεσα τα απαραιτητα αρχεια για να δουλεψει το git/reposerviceκαι δεν θα ασχοληθουμε με αυτα.  
  
Πηγαινουμε λοιπον ενα directoryπισω και εκτελουμε

**reposync**

**ΠΡΟΣΟΧΗ.  
Το reposync εκτελειται παντοντε στο ιδιο directory με τον φακελο .repo  
Αν καταλαθως εκτελεστει η εντολη μεσα σε καποιον φακελο ,θα υπαρχει προβλημα σε ολο το source**



Μετα απο τη τελευταια μας εντολη , εχουμε κατεβασει ολο το sourcecodeστον υπολογιστη μας.

Μπορουμε επισης να κατεβασουμε μονο το 1 repositoryεαν θελουμε κανοντας

**reposyncmyTestRepoLocalDir**

Αυτοθακανει μια ανανεωση το τοπικο sourcecodeμε 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 reposync για το repository που θελω  
  
Το reposyncμπορει να παρει ως ορισμα τα threadsμε τα οποια θα κανει sync.

**reposync -j4**

Γιατι ομως να μας ενδιαφερει το ποσα threadsτρεχουν κατα το reposyncαπο τη στιγμη που γινονται ολα μεσω ιντερνετ ?

Γιατι το repo ,κατεβαζει το sourcecodeσε συμπιεσμενη μορφη,αποσυμπιεζει τα αρχεια και μεσω μιας διαδικασιαςdiffcheckελεγχει ποια αρχεια απο το sourceμας εχουν αλλαξει ωστε να τα αντικαταστησει.  
Αν εχουμε προβλημα με το μεγαλυτερο -jαπλα το μειωνουμε.  
  
Σημειωση , το -jεπιρρεαζει και τις εργασιες που κανει το δικτυο κατα το κατεβασμα.Πρακτικα ,μεγαλυτερο jσημαινει 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 commitκαι το πρωτο push

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

* 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 3η περιπτωση.

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

Εστω οτι θελουμε να κανουμε pushμια 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 committitleβαζω “src:mylibSwitch 2 varsfrominttodouble”.Παρολαυτα υπαρχει και η αλλαγη του ονοματος της class και θα γινει και αυτο push.Κατι το οποιο δεν θα θελαμε.*

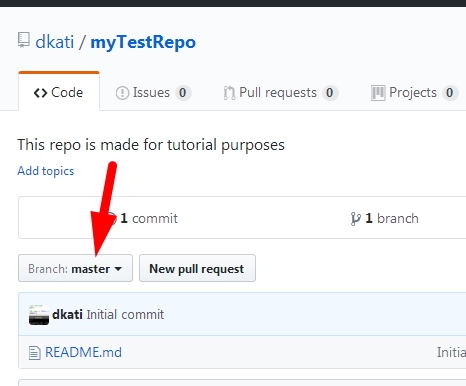
Πηγαινουμε λοιπον να κανουμε την πρωτη μας αλλαγη ,εστω στο myTestRepo  
Ο τοπικος φακελος του repositoryαυτου,ειναι το myTestRepoLocalDir.Οποτε

**cdmyTestRepoLocalDir**

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

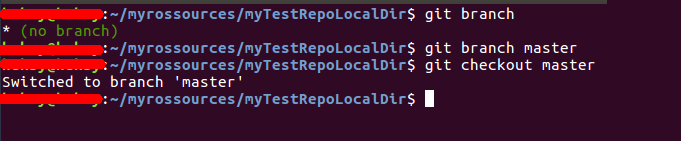
Και λογικα μας εμφανιζει (nobranch)

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



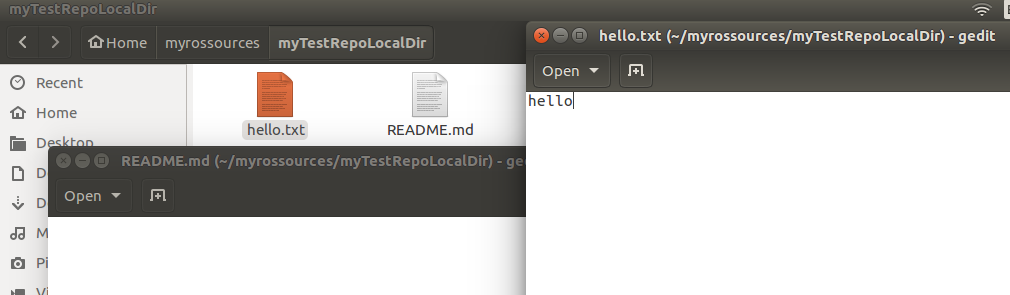
Aν θελω να στειλω τις μελλοντικες μου αλλαγες στο branchαυτο τοτε εκτελω

**git branch master  
git checkout master**

‘Η πιο απλα**gitcheckout -bmaster**

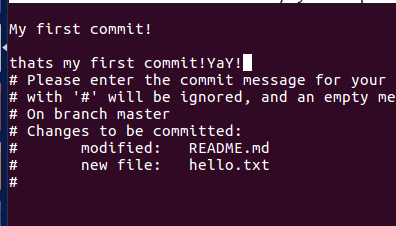
Απο τη στιγμη που αλλαξα το branchμου,μπορω ειτε να κρατησω το τερματικο ανοιχτο ειτε να το κλεισω.Δεν υπαρχει κατι που τρεχει στο background.

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 readmeοτι περιεχει



Εστω οτι αυτη ειναι η αλλαγη που θελαμε να κανουμε.Τωρα μενει να το προσθεσουμε στο commit.  
  
**gitadd -A  
gitcommit**

Αυτοματως θα μας εμφανισει ενα παραθυρο του nanoοπου κανουμε επεξεργασια το commitμας.  
Η πρωτη σειρα ειναι παντα το committitle.Αφηνοντας μια σειρα και προσθετοντας μια τριτη,γραφουμε την περιγραφη.

  
Παρατηρουμε τις δυο σειρες  
**modified: README.md  
newfile: hello.txt**

Αυτο μας επαληθευει οτι δεν «τραβηξαμε» στο commitκαποιο αρχειο που δεν θα επρεπε να ειναι μεσα.  
Αν ειναι ολα σωστα,τοτε συνεχιζουμε.

*Διαφορετικα παταμε ctrl+xκαι μετα Y χωρις να εχουμε συμπληρωσει committitleκαι description για να βγουμε.Πηγαινουμε στο φακελο που ειναι το .repo (δηλαδη στο root του myrossources) και ειτε διαγραφουμε το φακελο και κανουμε παλι reposync 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 reposync και μετα paste τα αρχεια που ειναι επεξεργασμενα*

Παταμε ctrl+Xκαι Yγια να κλεισει το commit.Βλεπουμε στο τερματικο οτι δειχνει το τιτλο του commitκαι αναφερει οτι 2 αρχεια αλλαξαν,1 προσθηκη και 2 διαγραφες .Οι προσθηκες και οι διαγραφες αφορουν ΣΕΙΡΕΣ κωδικα.Επισης μας δειχνει οτι δημιουργηθηκε το hello.txtμε mod 0644

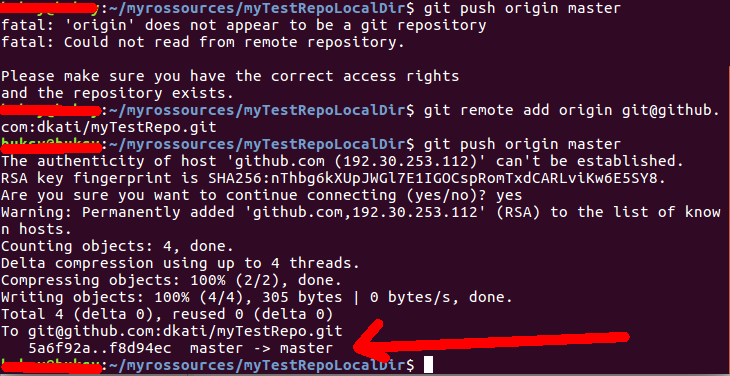
Απο τη στιγμη που εγινε το commit,πρεπει να το κανουμε push.

**gitpushoriginmaster**

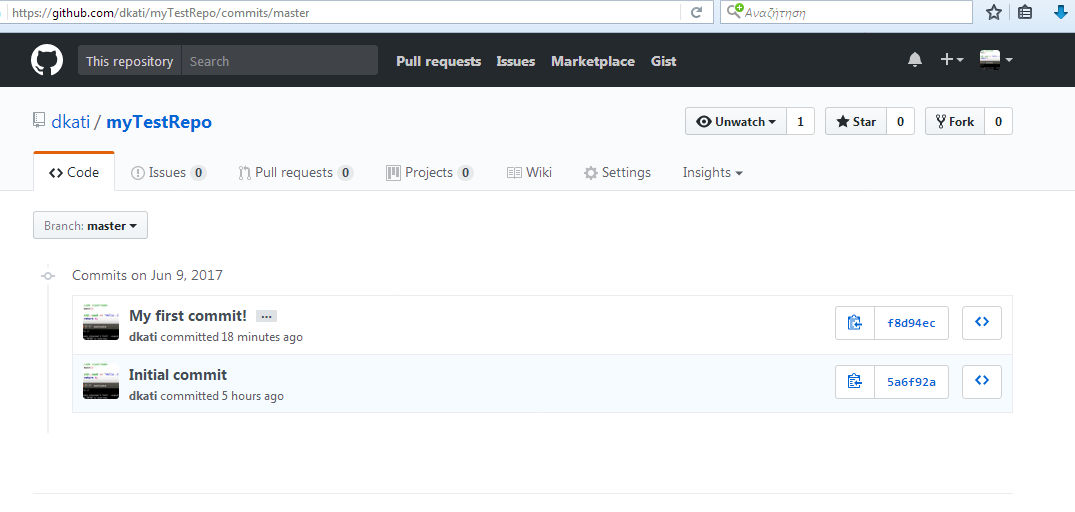
Η εντολη αυτη θα προσπαθησει να στειλει το commitστο branchmaster.  
Παρολαυτα μας βγαζει καποια fatalerrors.Ο λογος ειναι οτι χρειαζεται να φτιαξουμε (απαιτειται μονο μια φορα) το remote.Toremoteειναι υπευθυνο για την αποστολη του τοπικου commit,στο προφιλ μας.Συνεπως εκτελουμε (μια φορα και μονο φορα)

**git remote add origin** [**git@github.com:dkati/myTestRepo.git**](mailto:git@github.com:dkati/myTestRepo.git)

Καιξανακανουμε**git push origin master**

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

Στα commitsλοιπον του project ,πλεον βλεπω την αλλαγη μου



Κανοντας κλικ πανω στο commitβλεπω τις λεπτομεριες.



Παρατηρουμε ολα τα στοιχεια του commit.Το τιτλο,τις αλλαγες μας και την περιγραφη.

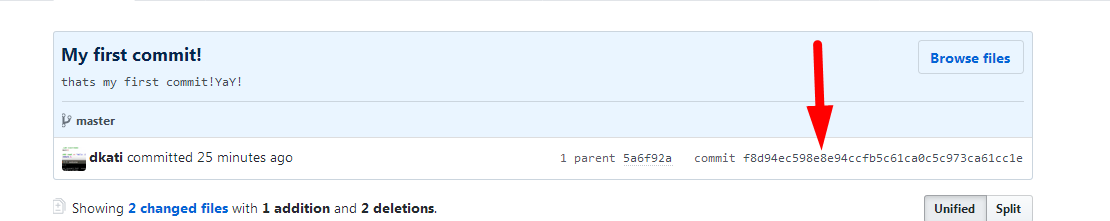
Οι κοκκινες γραμμες ειναι οτι σβηστηκε απο το αρχειο ενω οι πρασινες ειναι αυτες που εχουν προσθεθει.Αν ολες οι γραμμες απο ενα αρχειο ειναι πρασινες ,σημαινει πως το αρχειο εχει δημιουργηθει κατα το commit.

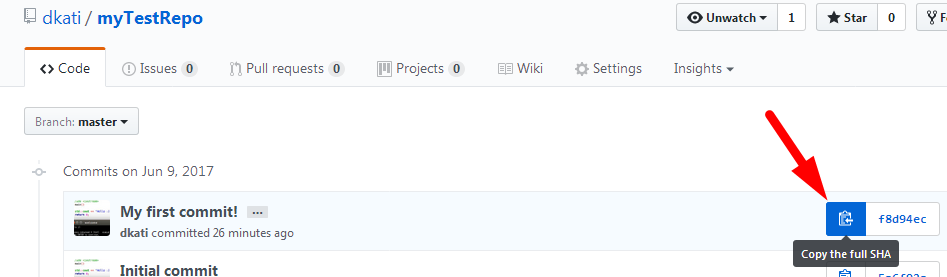
# αντιστροφη ενοσ commit

Εχουμε ηδη αναφερει πως στο githubδεν μπορουμε να σβησουμε commits.Οτι κανουμε pushπαραμενει στο ιστορικο.Παρολαυτα μπορουμε να αντιστρεψουμε ενα commit και αυτο γινεται με μια μονο εντολη.

**gitrevertSHA\_CODE**

ΤοSHAειναι το μοναδικο κλειδι που εχει καθε commitκαι το βρισκουμε μεσα απο το commitή απο το ιστορικο.

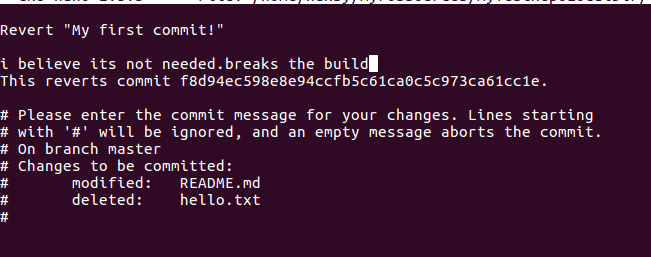


Οποτε ειτε το παιρνουμε με copypasteαπο εκει ειτε με ενα απλο κλικ ,οπως φαινεται στην κατω φωτογραφια  


Αυτο θα μας αποθηκευση στο clipboard,το commitSHA

Αρατρεχουμε**gitrevert7fb76ff5c019adf04bf2369583fbcee346a1c4dd**

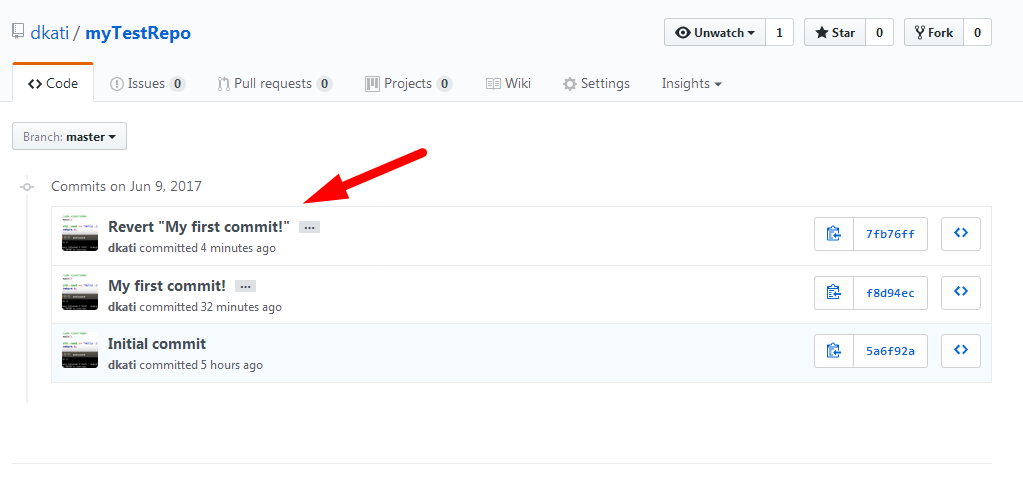
Οταν κανουμε το revertμας ανοιγει και παλι το commitmessage.  
Συνηθιζεται να αφηνουμε το committitleοπως ειναι,και να αλλαζουμε το descriptionεξηγωντας γιατι το καναμε αντιστροφη.  
αφου κανουμε τις απαραιτητες αλλαγες βγαινουμε μαζι με ctrl+xκαι y



Καικανουμετοpush

Git push origin master

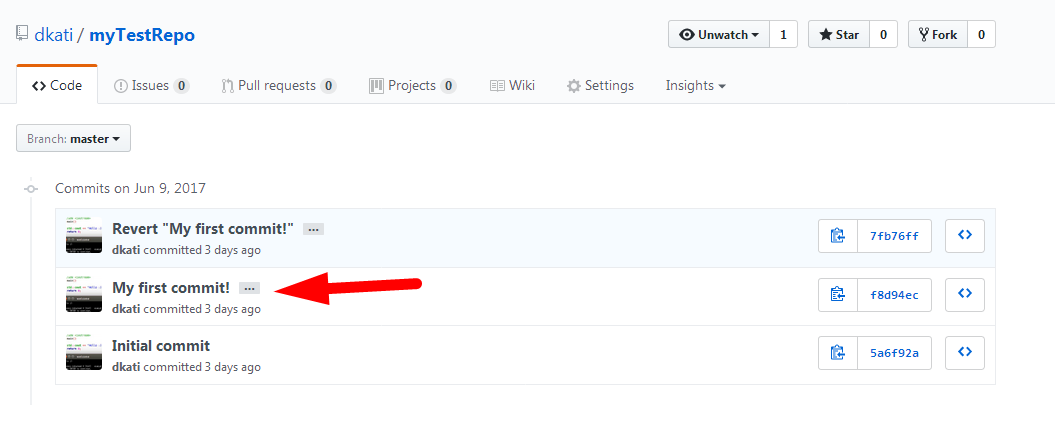
Αφου ολοκληρωθει το push ,μπορω να δω το commitστο commithistory



# cherry-picking.Η «νομιμη» αντιγραφη κωδικα

Οποιοσδηποτε κωδικας στο githubμπορει να ασφαλιστει με καποιο διεθνες license(apache,MITκτλπ).  
Τα licenseπροστατευουν τον συντακτη απο τυχον «κλοπες» του κωδικα.Το githubεχει μεριμνησει για την εξασφαλιση της νομιμοτητας μεσω των λειτουργιων του,και δινει τη δυνατοτητα στο χρηστη να αντιγραψει ενα commitκαποιου τριτου προγραμματιστη ,δινοντας τα απαραιτητα credits.  
Το cherry-pickμας 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 sourcecode,μαζί με τα commits,απο το οποιο θελουμε καποια συγκεκριμενα commits.  
Εστω οτι θελουμε να κανουμε cherry-pickτο commitπου καναμε πιο πριν “myfirstcommit” απο το   
<https://github.com/dkati/myTestRepo>

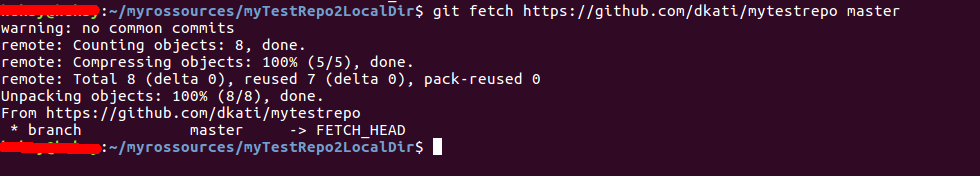


Αφου επιλεξαμε το commitπου θελουμε πρεπει να παμε μεσω του τερματικου στο directoryτου projectπου θελω να βαλω το commit.Εστω οτι θελουμε να μπει στοmyTestRepo2LocalDir.Eκτελουμε

**git fetch https://github.com/dkati/myTestRepo master**

Η συνταξη της εντολης ειναι   
gitfetch<linkτου προτζεκτ που θελουμε><ονομα branch>

Με την παραπανω εντολη κρατησαμε το ιστορικο των commitsτου συγκεκριμενου repository.



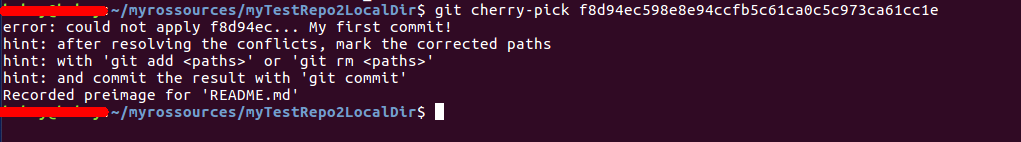
Πλεον ειμαστε ετοιμοι να «τραβηξουμε» το commit

**gitcherry-pickf8d94ec598e8e94ccfb5c61ca0c5c973ca61cc1e**

Η συνταξη της εντολης ειναι

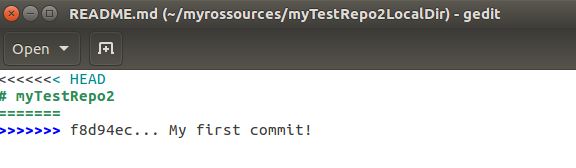
Git cherry-pick <SHA code>

Τρεχοντας την εντολη μας βγαζει errorπου αναφερει οτι δεν μπορει να προσαρμοσει το commitμεσα στον κωδικα μας.  
Οταν κανουμε cherry-pick,κατι τετοιο ειναι πολυ συχνο και οφειλουμε να ειμαστε σε θεση να το διορθωσουμε.Το αποτελεσμα της εντολης ειναι το παρακατω



Κοιτωντας το commitπαρατηρουμε οτι οι αλλαγες ειναι σε 2 αρχεια.



Παρολαυτα στο τερματικο μας βγαλε   
**RecordedpreimageforREADME.md(βλ.παραρτημα\*\*\*)**. Αυτο σημαινει οτι ΜΟΝΟ σε αυτο το αρχειο υπαρχει προβλημα.  
Ανοιγοντας το αρχειο βλεπουμε το εξης

Ας αναλυσουμε τη συνταξη του αρχειου

**<<<<<<<<< HEAD  
# myTestRepo2  
=======**  
Αυτο σημαινει πως κανονικα το αρχειο μας περιεχει οτι βρισκεται αναμεσα στο  
<<<<<<<< και το ======  
  
Απο το ====== μεχρι το >>>>> ειναι αυτο που περιεχει η αλλαγη του commit.  
  
Στη συγκεκριμενη περιπτωση το περιεχομενο αναμεσα στο ===== και στο >>>>ειναι κενο.  
αυτο σημαινει οτι το commitπου κανουμε cherry-pickσβηνει οτι περιεχεται απο το <<<<  
ως το ==== (το HEADδηλαδη).Συνεπως η σωστη λυση εδω ειναι να σβησω τα παντα.

Αποθηκευω το αρχειο και αφου πλεον εχουμε κρατησει τις αλλαγες μας ξεκιναμε το gitcommitμας

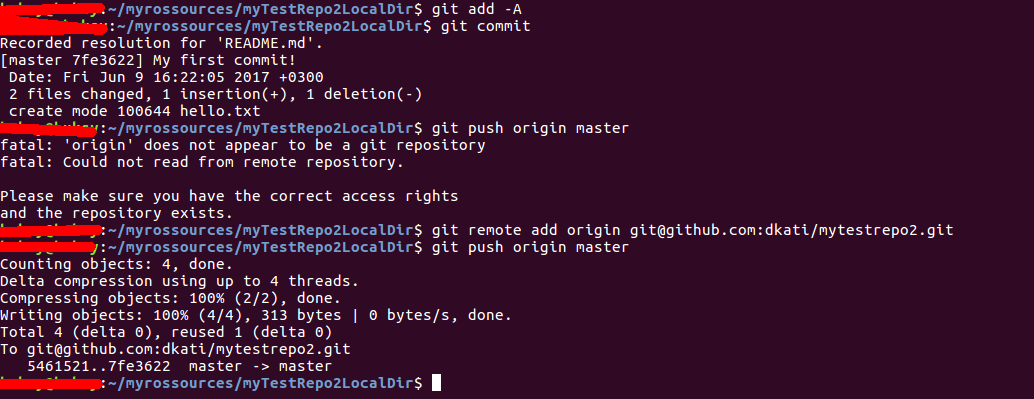
**gitadd -A  
gitcommit**  
  
και παρατηρουμε οτι εχει ηδη συμπληρωσει το committitleκαι το description.Παταμε ctrl+xγια να βγουμε απο το nano.Πλεον μπορουμε να κανουμε pushτο commit.

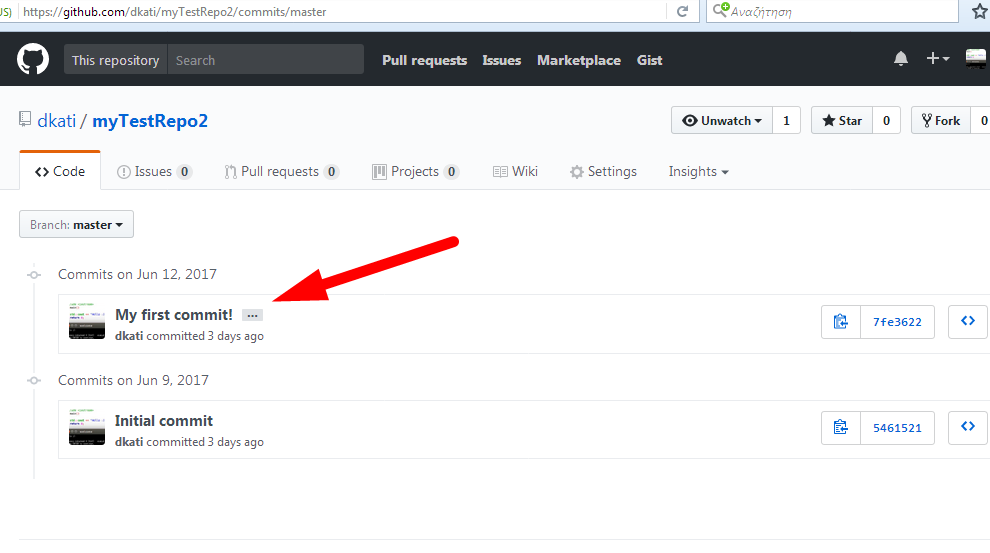
**gitpushoriginmaster**

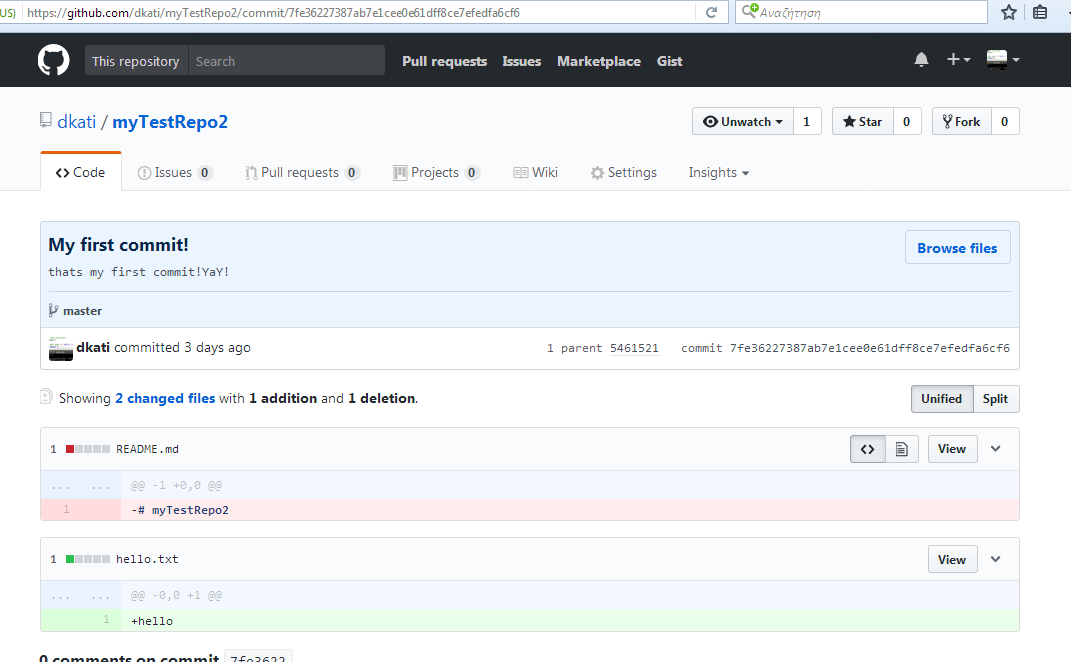
Στη περιπτωση μας θα μας βγαλει προβλημα στο remote.Οποτε οπως και πριν

**git remote add origin** [**git@github.com:dkati/myTestRepo2.git**](mailto:git@github.com:dkati/myTestRepo2.git)

Καιξαναgit push origin master



Πραγματι παρατηρηρουμε οτι το commitεγινε επιτυχως   




Αυτο που παρατηρουμε στα 2 commitειναι οτι αλλες αλλαγες εγιναν στο commitτου myTestRepoκαι αλλες εγιναν στο commitτου myTestRepo2.  
**Αυτος ειναι και ο λογος που ειχαμε conflict κατα το cherry-pick**.Ειναι σπανιες οι φορες που το cherrypickθα γινει χωρις conflict

Οτανεναcommitγινεταιcherry-pickκαι ο συντακτης ειναι διαφορετικο ατομο απο αυτον που εκανε το cherry-pickτοτε το commitεμφανιζεται ετσι



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

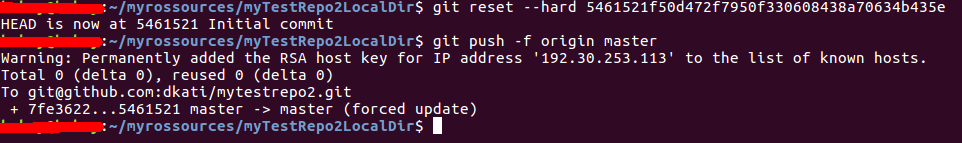
# Επανεγγραφητηςιστοριαστωνcommit(advanced users)

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

*Πρακτικα δεν τα διαγραφει την υπαρξη τους απλα δεν φαινονται στο commithistory.Επισης αναιρουνται απο τον τοπικο sourcecode.Η διαδικασια αναιρεσης της διαγραφης ειναι δυσκολη και απαιτειται μεγαλη εξοικειωση με το githubshell.*

Η συνταξη της εντολης ειναι

**gitreset –hardSHA\_CODE**(Διπλές παύλες)  
ΠχΑνθελωνα «σβησω» τοτελευταιοcommitτουmyTestRepo2 ,τοτεαρκεινακανωresetστοπροτελευταιο.  
**gitreset –hard 5461521f50d472f7950f330608438a70634b435e**  
καιμεταοφειλωνατοκανωpush  
**gitpush -foriginmaster**  
Το -fσημαινει «force» .Τοgithubτοκανειαυτογιαασφαλεια.ανδοκιμασουμενακανουμεgitpushoriginmasterδενθαμαςαφησει.

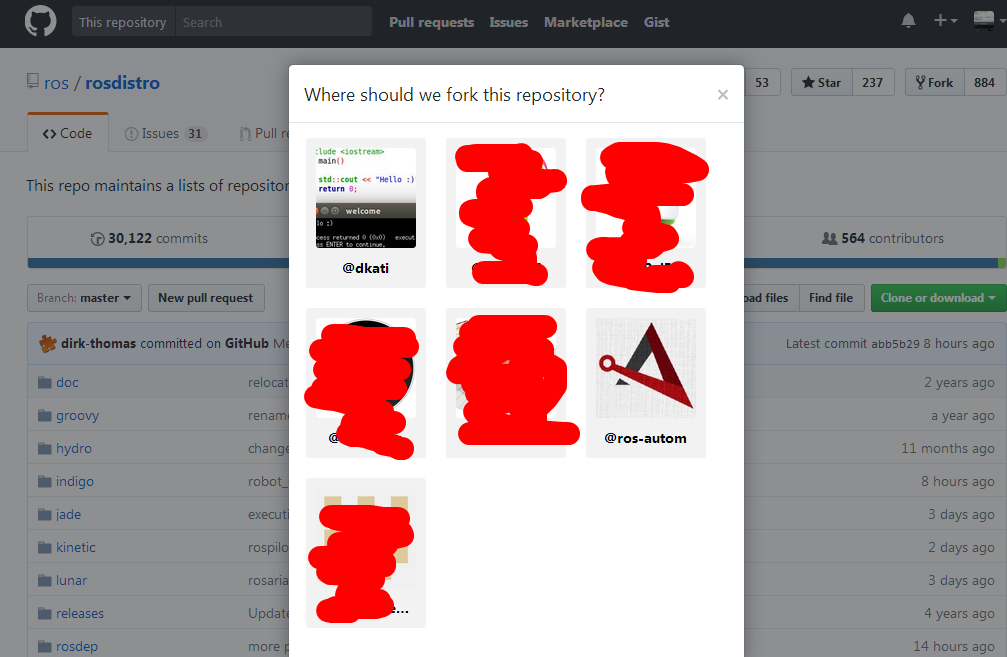


Αν το μετανιωσουμε μπορουμε να παρουμε το σβησμενο commit (ειναι το **7f3622**) κανοντας παλι reset  
  
**gitreset –hard 7f3622**

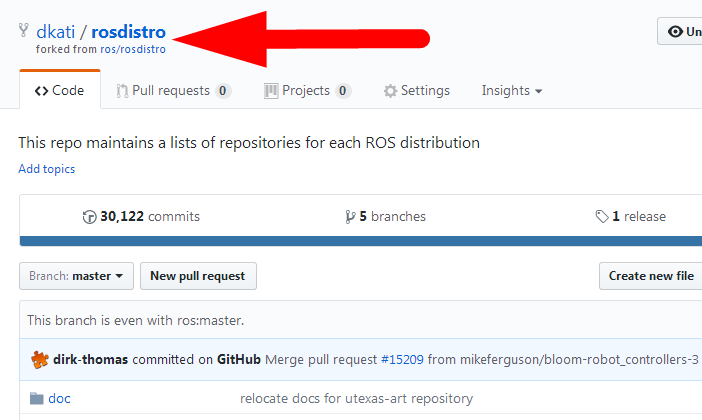
Συνεπως καταλαβαινουμε οτι το resetγραφει την ιστορια ειτε προς τα πισω ειτε προς τα εμπρος.

# Forking.Ο νομιμοσ τροπος να αντιγραψουμε ολοκληρα repositories

Forkingειναι η διαδικασια που αντιγραφουμε πληρως ολοκληρο το repositoryκαποιου.Ειναι πολυ απλο και επιτυγχανεται με 2 απλα click.

Πηγαινουμε στο repositoryτο οποιο θελουμε να κανουμε fork.Εστω το github.com/ros/rosdistro  
Παταμε πανω δεξια fork και επιλεγουμε σε ποιο λογαριασμο μας θελουμε να παει.  


Το repositoryεμφανιζεται στο προφιλ μας οπως παρακατω.  
φαινεται πολυ καθαρα οτι το repositoryειναι εργο του ros/distro

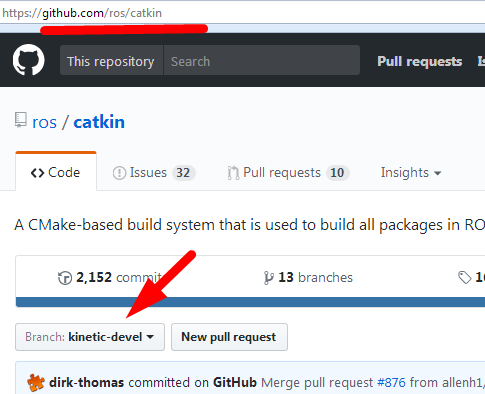


Απο εκει και περα χρησιμοποιω το projectοπως ακριβως εργαστηκαμε με τα αλλα 2 repositoriesτων παραδειγματων.

Παντοτε πρεπει να θυμομαστε οτι κατα το πρωτο pushπου θελουμε να κανουμε , χρειαζομαστε gitremoteaddorigin

# Γραψιμο ολοκληρης ιστοριασ απο αλλο repository

Εστωοτιεχωεναrepositoryμεbranchmasterκαι θελω να φτιαξω ενα ολοκληρο branchτο οποιο θα ειναι πρακτικα,το branchενος τριτου repository.  
Εστω λοιπον οτι θελω στο mytestrepoνα φτιαξω ενα branchπου θα περιεχει το kinetic-develbranchαπο το ros/catkinrepository



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

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

Συνεπως μπαινουμε στο myTestRepoLocalDir

**git clone** [**https://github.com/ros/catkin**](https://github.com/ros/catkin) **kinetic-devel  
cd kinetic-devel  
git branch**

Παρατηρουμε οτι ειμαστε στο branchπου εχουμε κατεβασει.Απο το σημειο αυτο φτιαχνουμε το δικο μας branch.

**gitbranchmykinetic  
gitbranch**

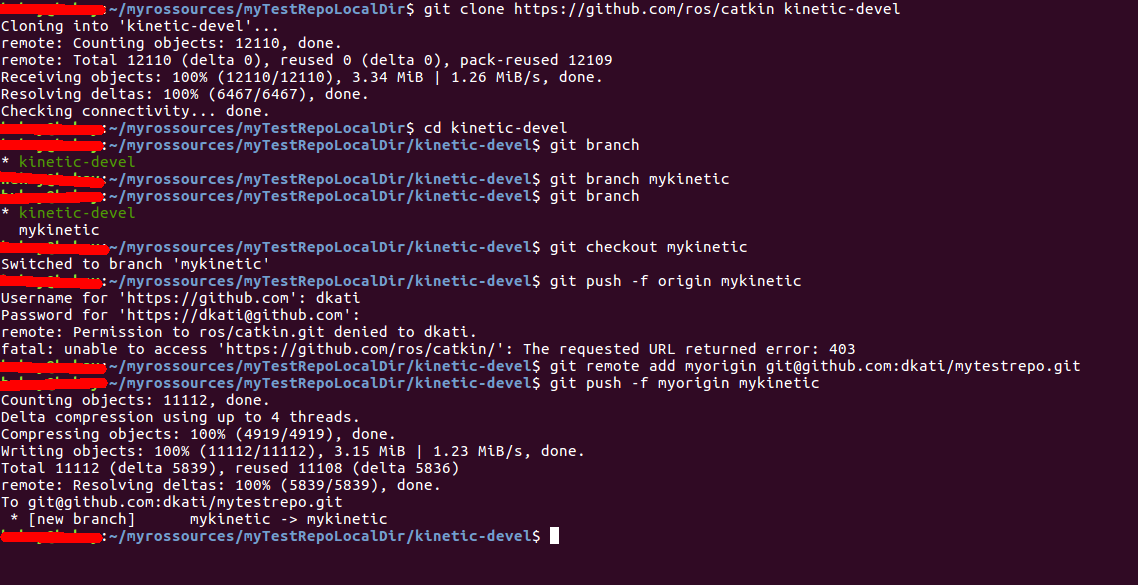
Πλεονβλεπωοτιεχω 2 branchesκαι ξερω οτι το mykineticεχει προερθει απο το kinetic-devel.  
Με πρασινο χρωμα ειναι το τοπικο branchστο οποιο βρισκομαστε.Αλλαζουμε λοιπον το branch

**gitcheckoutmykinetic**

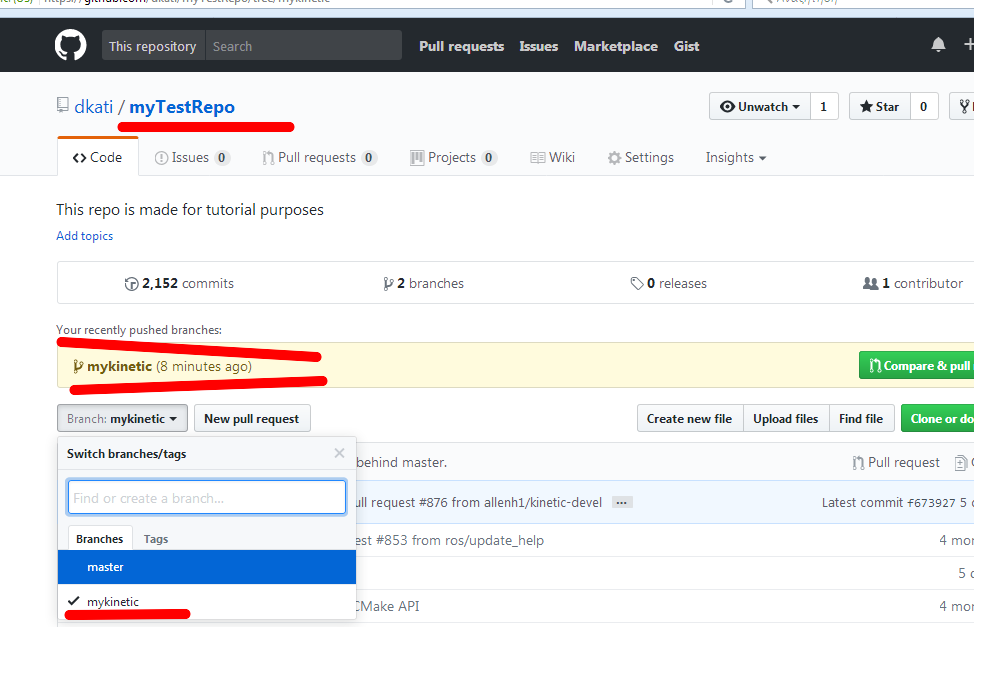
Αφου ειμαι στο branch,κανω pushοτι υπαρχει στο branch(και sourcecodeκαι commithistory).Αν γραψουμε Gitpushoriginmykinetic θα μας πει οτι δεν εχουμε δικαιωματα να κανουμε push το οποιο ειναι προφανες καθως δεν ανηκουμε στην ομαδα του ros.Αρα πρεπει να φτιαξουμε νεο origin

**git remote add myorigin** [**git@github.com:dkati/mytestrepo.git**](mailto:git@github.com:dkati/mytestrepo.git) **git push -f myorigin mykinetic**

Χρησιμοποιω -fγιατι αναγκαζω το githubνα μου φτιαξει στη σελιδα νεο branch



Εν τελει στο repositoryμας παρατηρουμε το νεο branchτο οποιο ειναι ΑΚΡΙΒΩΣ ιδιο με το πραγματικο branchτου ros

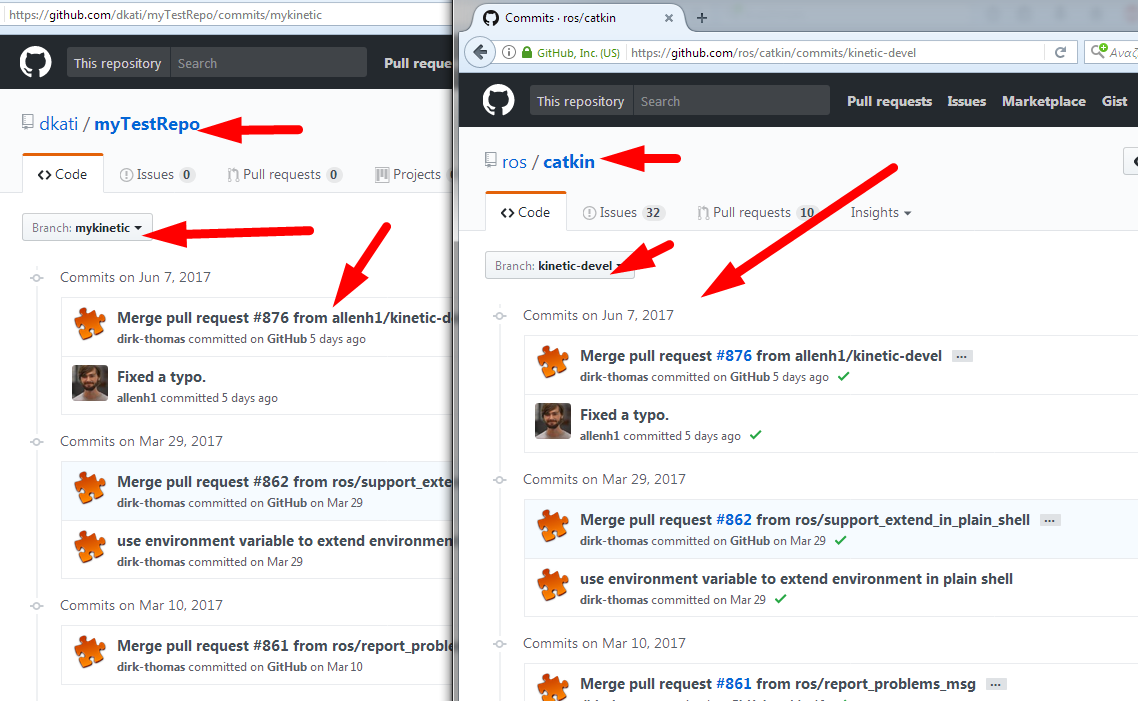


Αφου τελειωσαμε με το github,ας κανουμε ενα καθαρισμο τον κωδικα μας

**cd ../..  
rm -rf myTestRepoLocalDir  
repo sync myTestRepoLocalDir –force-sync**

--force-syncκαλο ειναι να χρησιμοποιουμε σε καθε reposyncγια να το αναγκασουμε να «τραβηξει» καθε νεα αλλαγη και να σβησει τυχον ξεχασμενα λαθη στο τοπικο source

Πολυ σημαντικο ειναι να παρατηρησουμε οτι ακομη και το commithistoryειναι ΑΚΡΙΒΩΣ ιδιο με το αυθεντικο repositoryτου ros



# Παραρτημα εντολων

* Git branch  
  Εμφανιση των διαθεσιμων τοπικων branches.Επισης εμφανιζει το branchστο οποιο ειμαστε τωρα(εμφανιζεται με πρασινα γραμματα).
* Git branch<branchname>  
  Δημιουργιαενοςνεουbranch ,αποτοbranchπου ειμαστε ηδη
* Git checkout <branch name>  
  Μεταβασησεαλλοbranch
* Git checkout -b<branchname>  
  Δημιουργιαενοςνεουbranchκαι μεταβαση σε αυτο
* Gitbranch -D<branchname>  
  Διαγραφηενοςbranch.ΔΕΝ ΜΠΟΡΟΥΜΕ ΝΑ ΔΙΑΓΡΑΨΟΥΜΕ ΤΟ BRANCHΣΤΟ 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> <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><branchname>  
  Κατεβασμα του commit history του repository.ΠΡΕΠΕΙ ΝΑ ΤΡΕΞΕΙ ΜΕΣΑ ΣΤΟ DIRECTORY ΣΤΟ ΟΠΟΙΟ ΘΑ ΚΑΝΩ ΤΟ cherry-pick
* repoinit -ugit://<link απο manifest>.git -bmaster  
  Αρχικοποιηση του manifest  
  πχ repoinit -ugit://github.com/dkati/myproject-manifest.git -bmaster
* gitcherry-pick<SHA>  
  Νομιμηαντιγραφηενοςcommitμε συγκεκριμενο SHA
* git reset –hard <SHA>  
  Επαναφορατουcommit history στοσυγκεκριμενοcommit
* gitrevert<SHA>  
  Αναστροφη του συγκεκριμενου commit
* git commit   
  Δημιουργια commit
* gitadd -A  
  Προσθηκη των αλλαγων μου στο commit
* git push <remote name><branch name>  
  Ανεβασματουcommit ,μεσωτουremote,στοbranch name
* git remote add <remote name>[git@github.com:<repo](mailto:git@github.com:%3crepo) name>.git  
  Δημιουργιαremote
* git remote remove <remote name>  
  Σβησιμοremote
* Repo sync ήrepo sync –force-sync  
  Συγχρονισμοςτοπικουsource code μεαυτονστοgithub. –force-sync ανθελωνααναγκασωτοgithub νασβησειτιςτοπικεςαλλαγεςμου

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

\*\*\*=  
Οταν το github μας αναφερει Recordedpreimageσημαινει οτι εχει καταγραψει το λαθος ΚΑΙ ΤΗΝ ΕΠΙΛΥΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ του χρηστη .Αυτο το κανει για να λυσει 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

**gitrerereforget \***

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