



**Εγχειρίδιο χρήσης GitHub   
για Windows (7 και πάνω)**

## Git και GitHub

Το **Git** είναι ένα ελεύθερο και ανοικτού κώδικα σύστημα ελέγχου έκδοσης που έχει σχεδιαστεί για να διαχειρίζεται τα πάντα από μικρά σε πολύ μεγάλα έργα προγραμματισμού και αρχείων με ταχύτητα και αποτελεσματικότητα.

Όταν είχε δημιουργηθεί το git , έτρεχε μόνο τοπικά, στον προσωπικό υπολογιστή του καθενός. Στη συνέχεια δημιουργήθηκαν προσωπικοί git servers ώστε ο προγραμματιστής/χρήστης να μπορεί να αποθηκεύει τις αλλαγές του εκεί. Ώσπου δημιουργήθηκε το **GitHub**. Ένας ελεύθερος και δημόσιος χώρος διανομής repositories που δίνει τη δυνατότητα δημιουργίας κρυφών repositories χωρίς τη χρήση ιδιωτικού server.

## Γιατί Git ;

Ο έλεγχος έκδοσης είναι ο μόνος λογικός τρόπος για να παρακολουθείτε τις αλλαγές σε κώδικα, χειρόγραφα, παρουσιάσεις και έργα ανάλυσης δεδομένων. Οι περισσότεροι χρησιμοποιούν zip αρχεία με διαφορετικές ημερομηνίες και εκδόσεις. Αλλά η διερεύνηση των διαφορών είναι δύσκολη, τουλάχιστον. Αν χρησιμοποιείται το git σωστά, η παραμικρή αλλαγή θα υπάρχει αποθηκευμένη και αρχειοθετημένη.

Η συγχώνευση των αλλαγών των συνεργατών γίνεται εύκολα. Έχετε ποτέ αντιμετωπίσει έναν συνεργάτη που σας έστειλε τροποποιήσεις που διανεμήθηκαν σε πολλά αρχεία ή έπρεπε να ασχοληθεί με δύο άτομα που έχουν κάνει αλλαγές στο ίδιο αρχείο ταυτόχρονα; Επώδυνος. *git merge* είναι η απάντηση.

## Γιατί GitHub ;

Όλοι είναι εκεί. Από Samsung και Apple, μέχρι Saga Robotics και TensorFlow project. Μπορείτε να δείτε τι δουλεύουν και να διαβάσετε εύκολα τον κώδικα τους, να κάνετε προτάσεις ή αλλαγές,να δείτε τον τρόπο που δουλεύουν και να παρατηρήσετε τις μικρές αλλαγές που κάνουν (commits)

Είναι Open source. Με το github, όλος ο κώδικας ελέγχεται εύκολα, όπως και ολόκληρο το ιστορικό του.Το Github μειώνει τα εμπόδια στη συνεργασία. Είναι εύκολο να προσφέρετε προτεινόμενες αλλαγές στον κώδικα των άλλων μέσω του github.

Δεν χρειάζεται να δημιουργηθεί ένα server. Είναι εκπληκτικά εύκολο να μπει κάποιος σε αυτό και να επεξεργάζεται κώδικα.

Τέλος, το github βοηθά στην ανάπτυξη των project ακόμη και απομακρυσμένα χωρίς να χρειάζεται να χρησιμοποιούμε τον προσωπικό μας υπολογιστή.

## Δεν ειμαι προγραμματιστής. Μπορώ να το χρησιμοποιήσω;

Το GitHub μπορεί να χρησιμοποιηθεί και για αποθήκευση αρχείων με ανάλογη χρήση όπως αυτή του dropbox,με τη διαφορά ότι στο github μπορείς να βλέπεις το ιστορικό των αλλαγών.  
Προσοχή! Το GitHub ΔΕΝ εξυπηρετεί ως file hosting server. Πέραν του μικρού χώρου που δίνει για αποθήκευση αρχείων, η φιλοσοφία του είναι πολύ διαφορετική.Φυσικά μπορεί να γίνει αποθήκευση αρχείων εκεί.

## Τι αρχεία υποστηρίζονται στο GitHub;

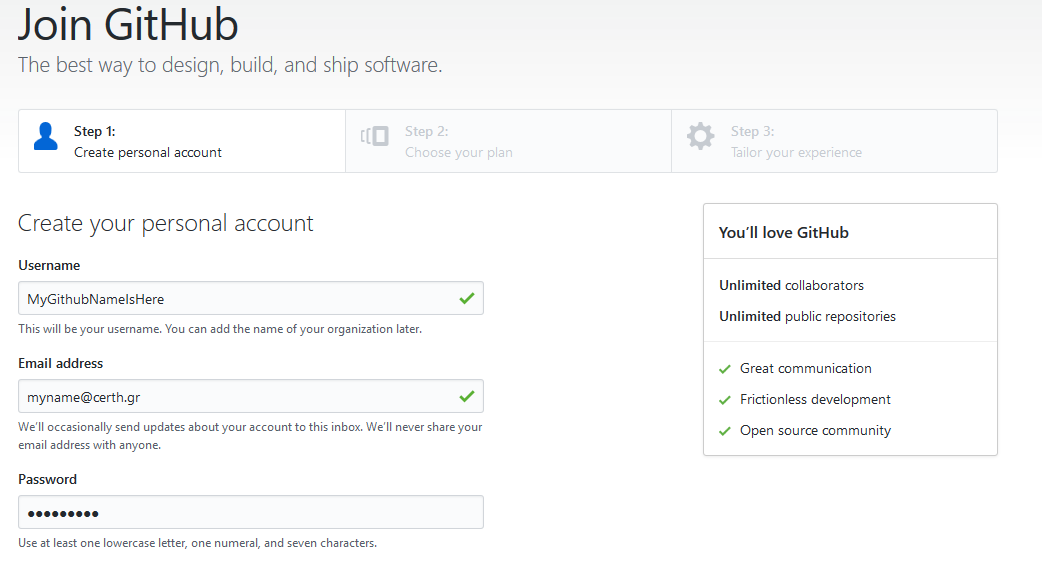
Όλα, με μέγιστο μέγεθος 100mb ανά αρχείο και 5GB ανά repository. Κάποια αρχεία παρόλα αυτά αναγνωρίζονται ως αρχεία που δεν μπορείς να δεις τις αλλαγές του μέσα σε αυτά.Παρακάτω ακολουθεί ένα σενάριο χρήσης GitHub με στόχο την βελτιστοποίηση ενός project προγραμματισμού.Πριν από αυτό όμως θα παρουσιαστούν η εγκατάσταση του GitHub για Windows, και μερικές ορολογίες χρήσιμες για την κατανόηση του τρόπου λειτουργίας του.

# Εγκατάσταση

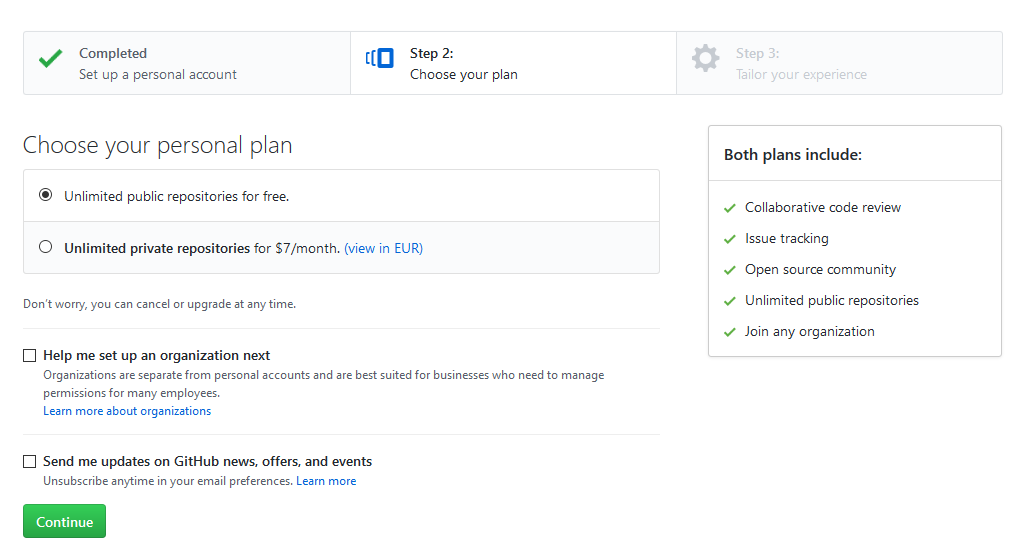
1. <https://desktop.github.com/>
2. Κατεβάζουμε το εκτελέσιμο.
3. Προχωράμε το setup το οποίο έχει αρκετά εύκολη πλοήγηση.
4. Όταν τελειώσει η εγκατάσταση προχωράμε στη δημιουργία λογαριασμού μέσω της σελίδας <https://github.com/join?source=experiment-header-dropdowns-home>
5. Ακολουθούμε τις οδηγίες όπως φαίνεται παρακάτω

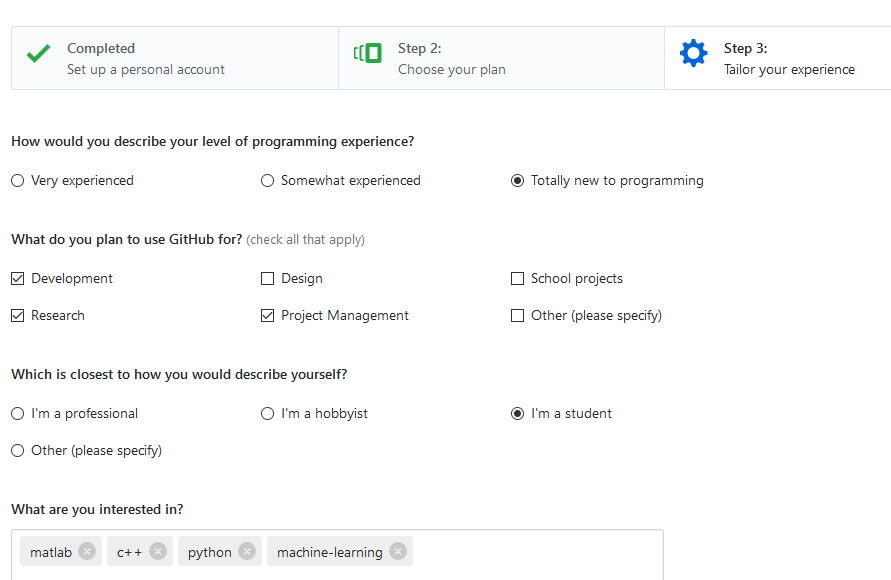
Ένα οποιοδήποτε email για να την επιβέβαιωση των στοιχείων. Μπορούν να προσθεθούν πολλά emails σε έναν λογαριασμό στη συνέχεια

Το όνομα θα φαίνεται δημοσίως και θα υπάρχει σε κάθε σας commit/αλλαγή που κάνετε στο project σας



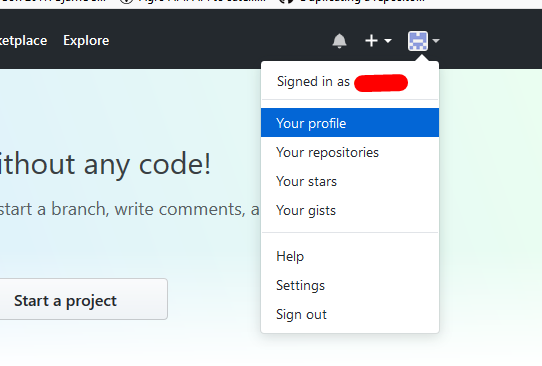
To iBO διαθέτει ήδη δωρεάν ιδιωτικά repositories





Επιλέγουμε ότι επιθυμούμε.Δεν είναι δημοσίως προσβάσιμα στοιχεία.

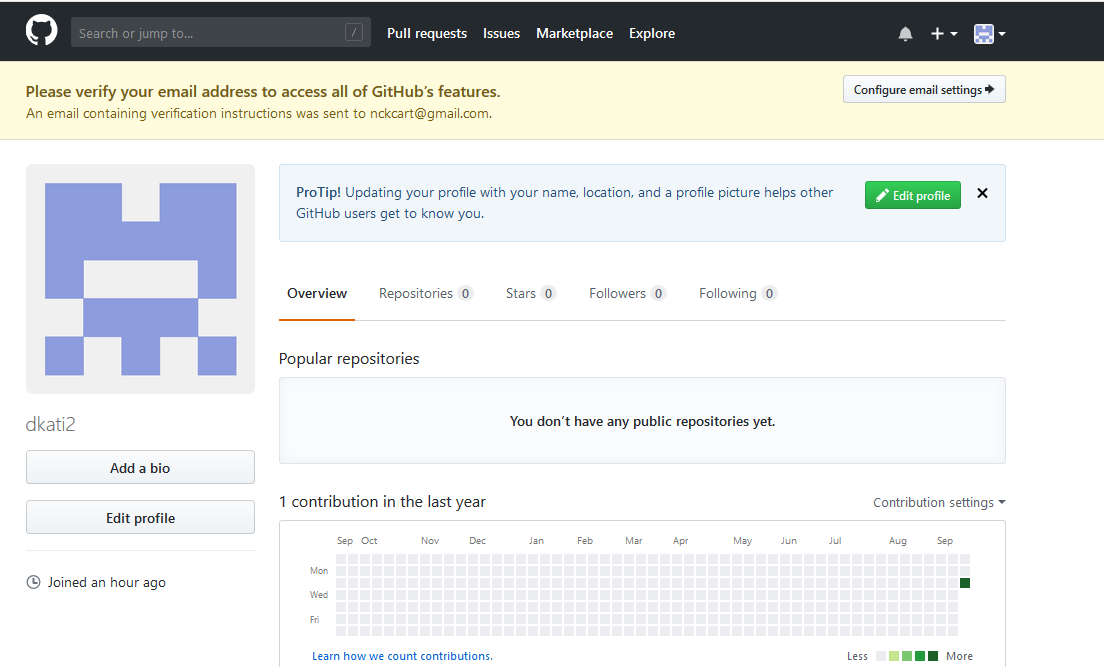
Στη συνέχεια πηγαίνουμε στο προφίλ πατώντας πάνω δεξιά στο avatar



Παρακάτω παρατηρούμε την κύρια σελίδα του λογαριασμού μας.

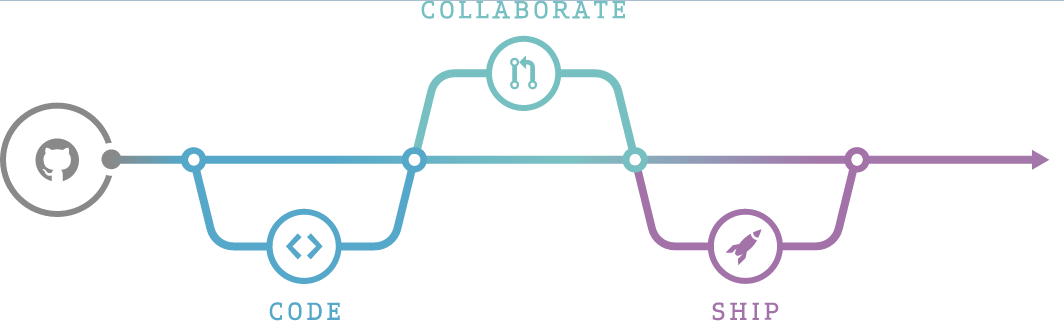
Ιστορικό των συνεισφορών μας

Πλοήγηση στα repositories



Από τη στιγμή που το προφίλ είναι έτοιμο, μπορούμε να δημιουργήσουμε οποιόδηποτε repository. Πριν από αυτό, θα προηγηθεί μία ανάλυση του τρόπου λειτουργίας του github.

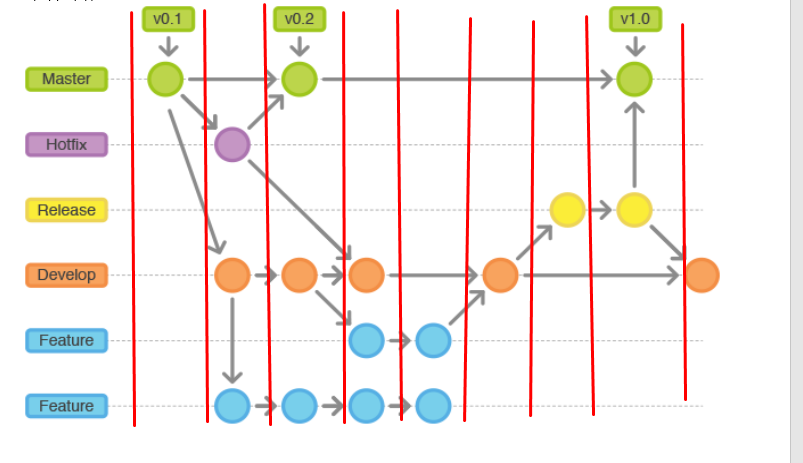
# Τρόπος λειτουργίας και η φιλοσοφία του GitHub



Τη λειτουργια του github μπορουμε να τη φανταστουμε ως ενα δεντρο (tree) με κλαδια (branches) και κυριο κορμο (master branch),πανω σε ενα χρονοδιαγραμμα.Για την ακριβεια ως ενα δεντρο με κερασια (Θα αναλυθει παρακατω).Παρακατω θα μελετησουμε ενα flowchart μιας τυχαιας 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 branch που 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 master branch.  
Αυτο σημαινει οτι ειναι το ΠΡΩΤΟ-initial release του source code μας στο github.Mπορουμε να φανταστουμε το master branch ως το «επισημο» source code που θα θελαμε καποιος να δει.Συνηθως ειναι συνήθως η stable εκδοση του τρεχοντα κωδικα.Φυσικά το κύριο branch μας μπορεί να ονομάζεται και διαφορετικά.

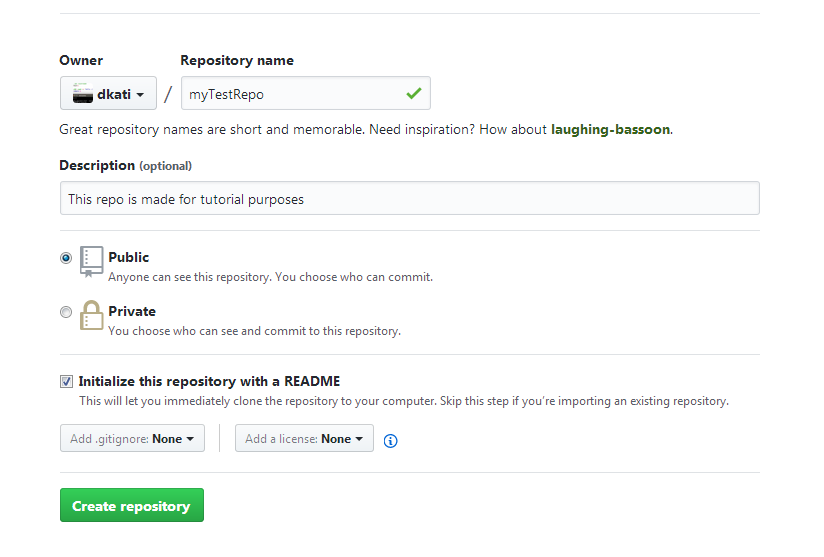
Στη συνεχεια παρατηρουμε 3 βελακια.  
1 προς το hotfix.1 προς το develop.1 προς το feature.  
  
Αυτο σημαινει οτι απο το master φτιαξαμε αλλα 3 branches.Βλεπουμε πως τα κυκλακια αυτα δεν ειναι στην ιδια καθετο με το master.αυτο σημαινει πως εχει γινει καποιο commit-καποια αλλαγη στον κωδικα.  
Συνεπως βλεπουμε οτι ο προγραμματιστης του κωδικα εφτιαξε 3 ακομα 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 παρακάτω*  
Στην τρίτη κάθετη γραμμή βλέπουμε το hotfix να μπαίνει στο master.Αυτο σημαινει πως το hot fix ηταν πιθανον ενα bug fix ,οποτε ο προγραμματιστης το προς΄σρμοσε στο κυριo branch.Επίσης ο προγραμματιστής ανεβάζει την εκδοση του προγραμματος σε v0.2

Στην ίδια τριτη κάθετη γραμμή παρατηρω και τα αλλα κυκλακια απο develop και 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

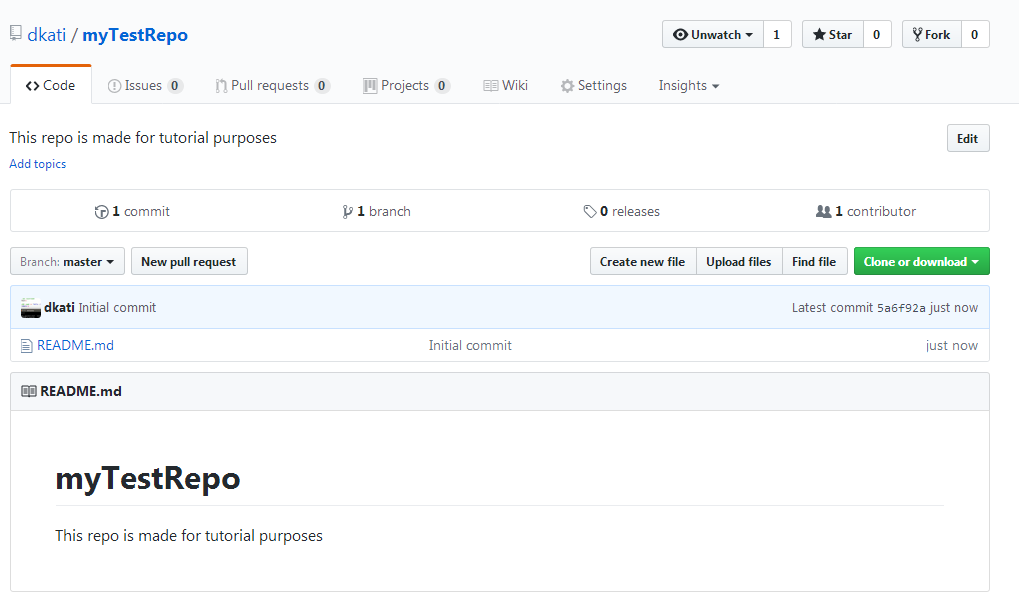
**Παρατηρήσεις**

* 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 σημαινει 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 και το commit history ονομαζεται repository (repo)
* Το οτι κανω push μια αλλαγη/ενα commit δεν σημαινει 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

# Ξεκινώντας νεο repository για να ανεβάσουμε τον κώδικα ενος project που έχουμε ήδη γράψει

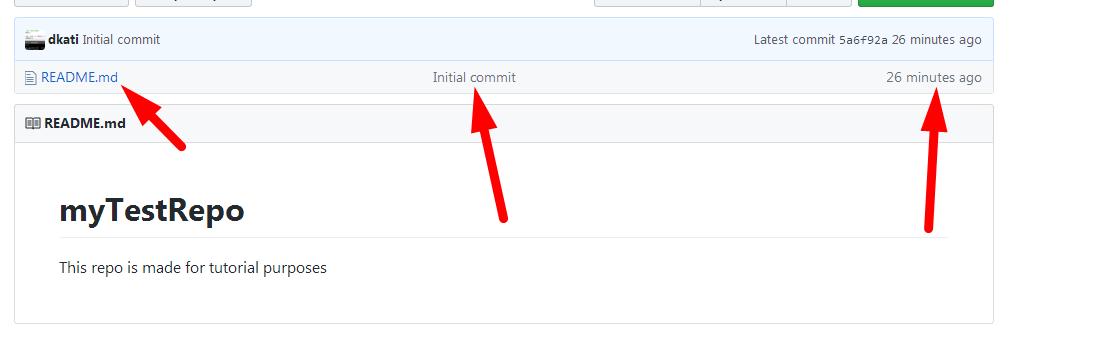
Μεταφερομαστε στο προφιλ μας στο github.com.Αφου εχουμε ηδη κανει login , επιλεγουμε το **+** απο πάνω δεξια και επιλεγουμε New repository.Εκει μπορουμε να συμπληρωσουμε τα στοιχεια του αρχικου repository.Αρχικα θα ειναι αδειο και μετα θα κανουμε προσθηκη του κωδικα μας ως initial release.Γραφουμε το ονομα του repository ,μια περιγραφη εαν θελουμε,Δημοσιο repository και επιλεγουμε και το Initialize this repository with a README για να δημιουργησει το πρωτο μας αρχειο.Αργοτερα θα προσθεσουμε και το license.  


Αφου πατησουμε Create repository θα 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
  + Pull requests : 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 contributors δινουν καποιες οδηγιες σχετικα με το source code
  + Settings : Ρυθμισεις του repository(Οχι του source 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.
* New pull request: Εαν θελω ως τριτος χρηστης να προσθεσω κωδικα
* Create new file/Upload files : Χειροκινητη δημιουργια/δημοσιευση αρχειου **(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 αρχειο μπορει να διαγραψει 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

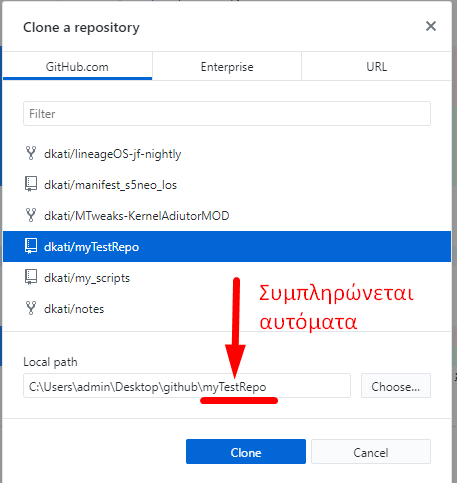
Το μεσαιο βελακι δειχνει το τελευταιο commitπου εχει γινει **και εχει επιρρεασει το συγκεκριμενο αρχειο**

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

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

# Συνδέοντας το repository με τα τοπικά μας αρχεία

Από τη στιγμή που το απομακρυσμένο repository έχει δημιουργηθεί, το μόνο που απομένει, είναι ο συγχρονισμός του με τα τοπικά μας αρχεία. Εκτελούμε την εφαρμογή που έχουμε κάνει εγκατάσταση. Συνδέουμε το λογαριασμό μας και προχωράμε.Επιλέγουμε από το μενού επιλογών File->Clone repository. Στο path που ζητάει καλό θα είναι να έχουμε έναν φάκελο κάπου τοπικά ώστε να μπορούμε να έχουμε όλα τα github project μας μαζεμένα σε ένα σημείο. Κατά το clone θα δημιουργηθεί ένας φάκελος μέσα στο path που θα επιλέξουμε. Συνεπώς δε χρειάζεται να φτιάξουμε και άλλον φάκελο μέσα σε αυτό για κάθε ξεχωριστό project. Επιλέγουμε το repository που μόλις φτιάξαμε.



Στη συνέχεια πατάμε clone και περιμένουν να κατέβει το απομακρυσμένο repository. Όταν ολοκληρωθεί, θα έχουμε μία εικόνα σαν την παρακάτω.

Επιλογή συγχρονισμού απομακρυσμένου repository με τοπικά αρχεία

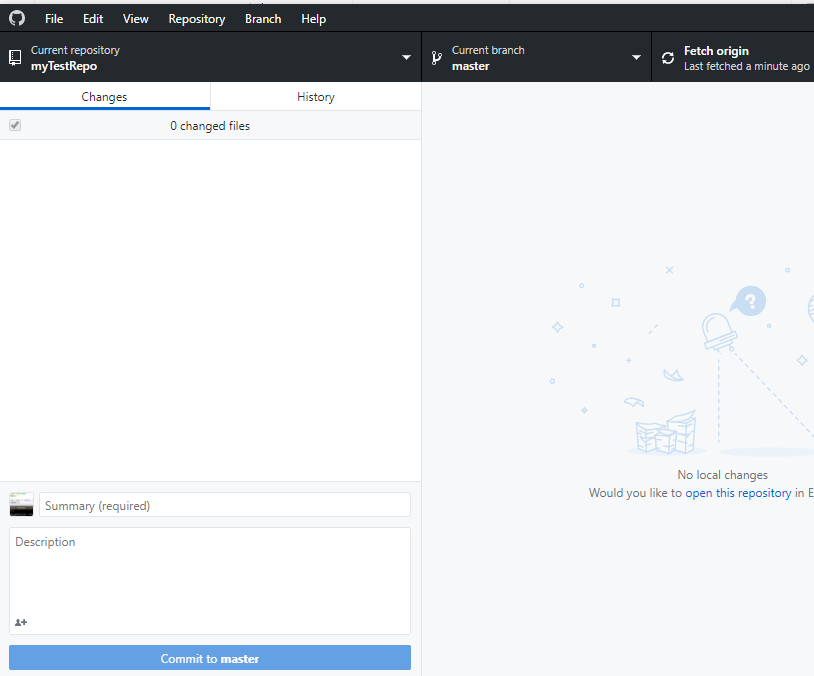
Τρέχον branch

Καρτέλα ιστορικού

Καμία αλλαγή προς το παρόν

Τρέχον repository

Ταμπέλα  
αλλαγών

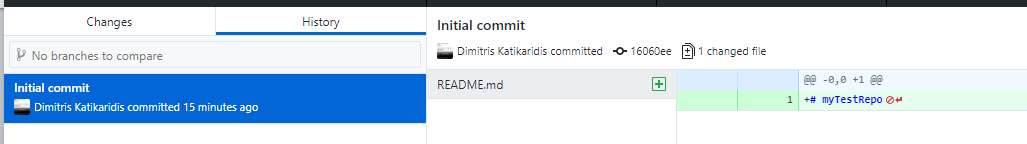


Πατώντας στο ιστορικό βλέπουμε την παρακάτω εικόνα

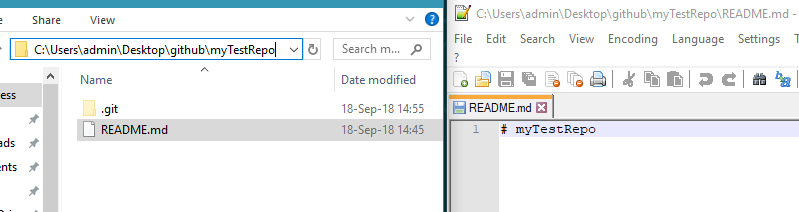
Αλλαγές που έγιναν στο αρχείο αυτό.

Αρχεία που επιρρεάστηκαν από το commit

Πληροφορίες σχετικά με το commit.Το συγκεκριμένο έγινε αυτόματα με τη δημιουργία του repository

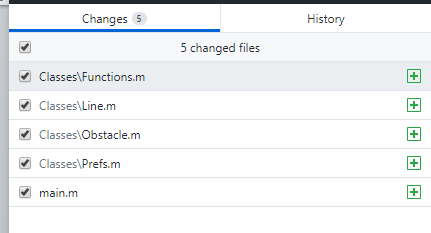
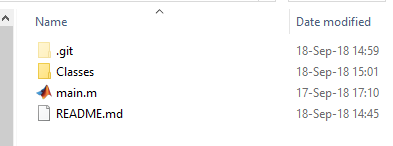


Αν πάμε στο path που δώσαμε κατα το clone του repository θα δούμε τα αρχεία από το απομακρυσμένο repository.



Έστω λοιπόν ότι θέλουμε να ανεβάσουμε στο repository το μέχρι τώρα project μας. Προσθέτουμε τα αρχεία μας μέσα στο φάκελο **myTestRepo** και μεταφερόμαστε στην εφαρμογή του GitHub.

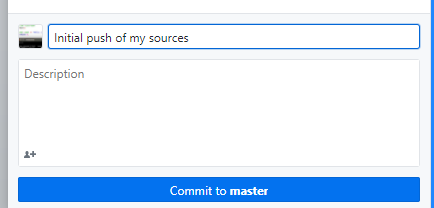
Ας υποθέσουμε ότι το project μας περιέχει έναν φάκελο με όνομα **Classes** και το αρχείο MatLab **main.m** . Πηγαίνοντας στην εφαρμογή του GitHub ,στην καρτέλα «Changes» , παρατηρούμε ότι η εφαρμογή έχει αναγνωρίσει τις αλλαγές μας όπως φαίνεται παρακάτω.



Επιλέγοντας κάθε αρχείο, βλέπουμε το τι έχει προσθεθεί. Φυσικά, από τη στιγμή που ανεβάζουμε ολόκληρο αρχείο, το github «αντιλαμβάνεται» ότι ανεβάζουμε καινούριες σειρές. Συνεπώς όλο το αρχείο είναι στην ουσία πολλές προσθήκες νέων σειρών.

Στη συνέχεια βάζουμε κάτω αριστερά ένα commit title και πατάμε το κουμπί «commit to master» .

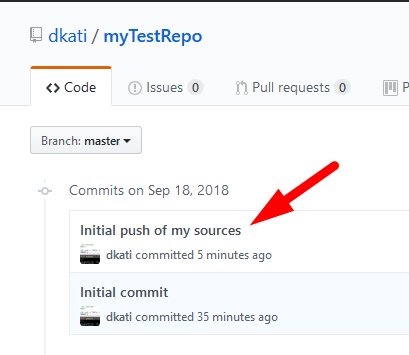
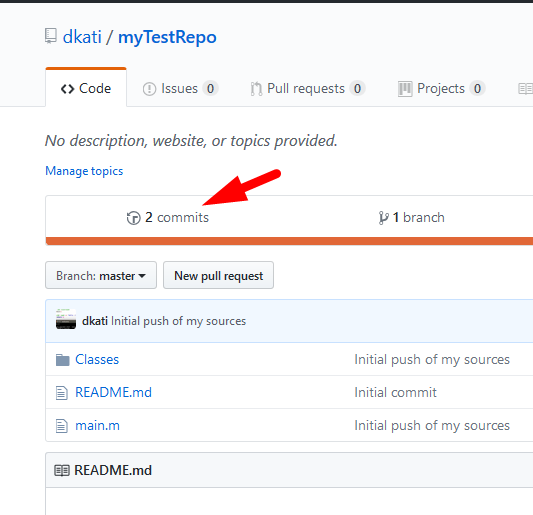
Φυσικά στην περιγραφή μπορούμε να βάλουμε ότι θέλουμε ή και τίποτα. Μόλις πατήσουμε το κουμπί, θα παρατηρήσουμε ότι αφενώς έχουν εξαφανιστεί οι αλλαγές από την εφαρμογή και αφετέρου το η επιλογή από την μπάρα επιλογών έχει αλλάξει από «fetch origin» σε «push origin».



Επίσης παρατηρούμε και ένα βελάκι με τον αριθμό ‘1’. Αυτό σημαίνει πως έχουμε 1 ΤΟΠΙΚΟ commit το οποίο πρόκειται να κάνουμε push στο απομακρυσμένο repository.

***Best practice : Κάνουμε push , ένα-ένα τα commits για να αποφύγουμε conflicts***

Πατώντας το push origin η εφαρμογή θα αποστείλει την αλλαγή μας στο GitHub. Όπως είδαμε και πριν, επιλέγουμε την καρτέλα “X commits” από τη σελίδα του repository για να δούμε το commit.



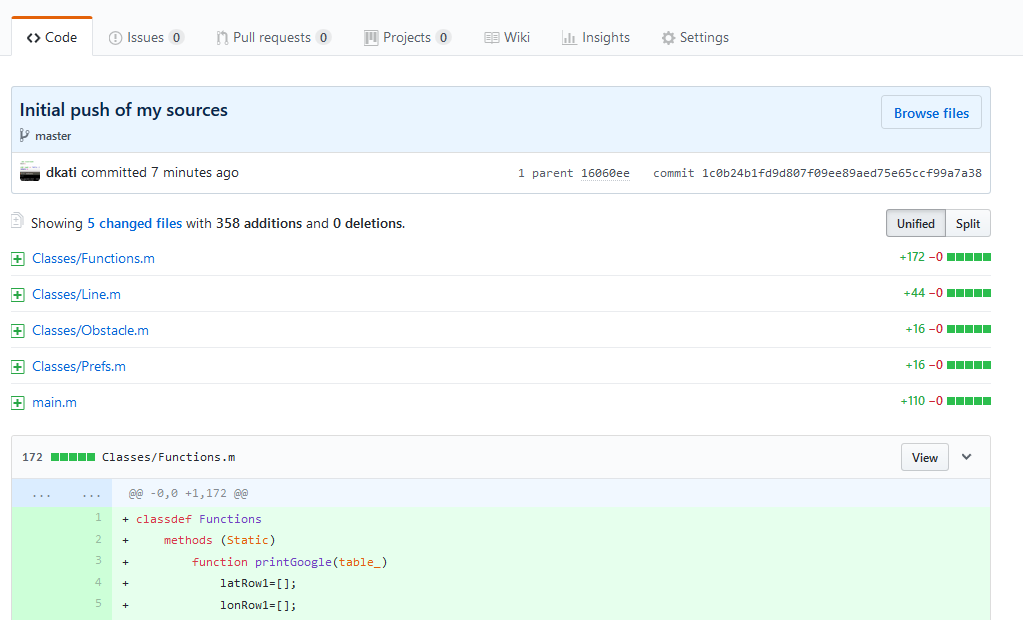
Πατώντας πάνω στο commit βλέπουμε τις αλλαγές που προκαλέσαμε. Όταν μπούμε μέσα στο commit επιλέγουμε το ‘X files changed’ για να δούμε ποια αρχεία επιρρεάστηκαν.

-Όνομα commit  
-Περιγραφή (αν υπάρχει)  
-Ποιο branch  
-Ποιος  
-Πριν πόση ώρα

Αριθμός νέων και αφαιρούμενων σειρών κώδικα του εκάστοτε αρχείου

Το σύμβολο + σημαίνει «νέο αρχείο»  
Αντίστοιχα ‘-‘ σημαίνει «διαγραφή αρχείου» και κίτρινη τελεία σημαίνει «τροποποίηση αρχείου»

Αριθμός νέων και αφαιρούμενων σειρών κώδικα



Με αυτόν τον τρόπο έχουμε πλέον όλα τα τοπικά μας αρχεία , συγχρονισμένα με το απομακρυσμένο repository.

# Το πρωτο πραγματικό commit και push

Το πιο σημαντικό σημείο είναι να καταλίβουμε τι θέλουμε να κάνουμε push και πότε θέλουμε να το κάνουμε push.Ένα commit πρέπει να ειναι καθαρό.Αυτο σημαίνει ότι το καθε commit πρέπει

* Να εχει ξεκάθαρο τίτλο που θα περιγράφει τι ακριβώς κάνει
* Αν είναι απαραίτητο να έχει μια περιγραφή με λεπτομέριες
* **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 νόμιμο τρόπο που θα δούμε παρακάτω.

Ας εξηγησουμε την 3η περιπτωση.

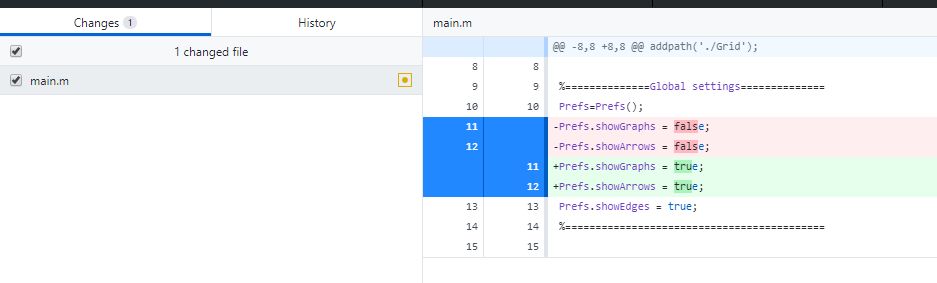
Κατα την εξοικιωση μας με το github,κανουμε push πραγματα τα οποια δεν πρεπει να γινουν push.

**Παράδειγμα-Σενάριο**  
 *Σε ενα αρχειο αλλαζω 2 μεταβλητες απο* ***false*** *σε* ***true*** *και στο ίδιο αρχείο αλλάζω το όνομα μιας συνάρτησης.*

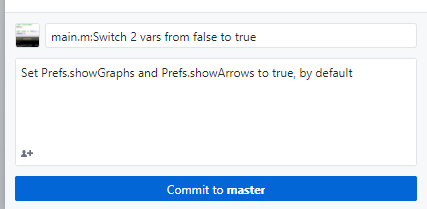
Στο commit title βαζω **“main.m:Switch 2 vars from false to true”** με περιγραφή **«Set Prefs.showGraphs and Prefs.showArrows to true, by default»** .Παρόλαυτα υπάρχει και η αλλαγή του ονόματος της συνάρτησης και θα γίνει και αυτο push.Κατι το οποίο **δεν** θα θέλαμε καθώς είπαμε πως το commit title πρέπει να περιγράφει ΠΛΗΡΩΣ την αλλαγή που προκάλεσε το commit.

Η σωστή κίνηση θα είναι να κάνω πρώτα την αλλαγή στις 2 μεταβλητές, να κάνω το push και στη συνέχεια,σε δεύτερο commit, να αλλάξω όνομα στη συνάρτηση.

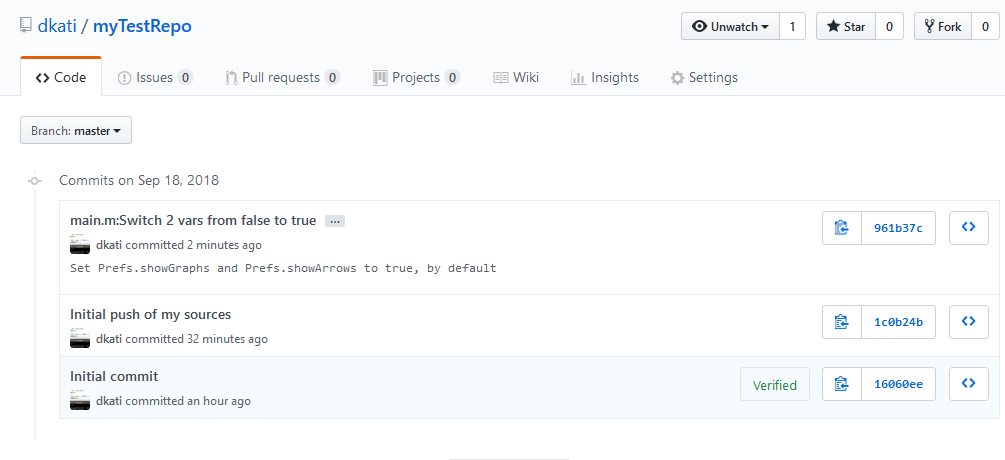
Κάνουμε την αλλαγή στα αρχεία μας και μεταφερόμαστε στην εφαρμογή του github.



Παρατηρούμε ότι έχει αναγνωρίσει τις αλλαγές μας. Όποτε συμπληρώνουμε σωστά το commit title και description επιλέγουμε το «commit to master» και κάνουμε το push πατώντας το κουμπι push πάνω από την εργαλειοθήκη.



Φυσικά το commit φαίνεται και στη σελίδα του github.

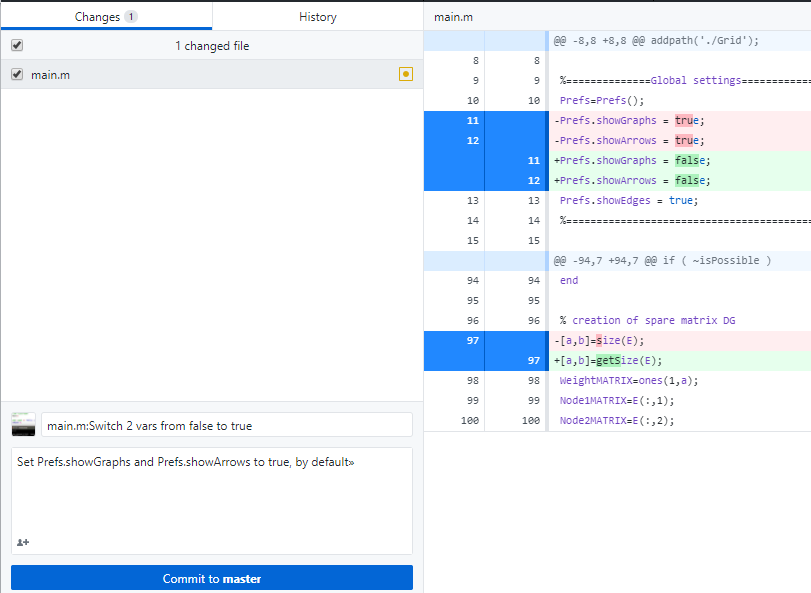


Πατώντας στις 3 γκρι τελείες μας εμφανίζεται το commit description.

Συνεπώς από εκεί και πέρα μπορούμε να κάνουμε μεταονομασία τη συνάρτηση που θέλουμε και να κάνουμε ένα ακόμη commit με τίτλο έστω **«main.m:Rename function1 to myfunction1»**

# Αντιμετώπιση του παραπάνω σεναρίου σε περίπτωση που κάνουμε καταλάθως ΚΑΙ την αλλαγή του ονόματος συνάρτησης μέσα σε ένα commit

Έστω ότι έχει γίνει αλλαγή και των μεταβλητών αλλά και μεταονομασία της συνάρτησης.



Παρατηρούμε ότι οι αλλαγές περιέχουν και την μεταονομασία. Επιλέγοντας τις 2 σειρές (που έχουν τον αριθμό 97) τότε τις βγάζω εκτός του συγκεκριμένου commit. Με αυτόν τον τρόπο μπορούμε να αφαιρέσουμε αυτήν την αλλαγή απο το συγκεκριμένο commit. Όταν γίνει το push τότε τις επιλέγω ξανά ώστε να τις κρατήσει στο νέο-δεύτερο commit που θα κάνω για την μεταονομασία.

# Παράρτημα-ορολογίες

Repository 🡪 Αποθετήριο κώδικα.περιέχει τα αρχεία μας μαζί με το ιστορικό αλλαγών και τα branches

Branch 🡪 Η διακλάδωση του project. Υπάρχει για να μπορούμε να έχουμε πολλές εκδόσεις του ίδιου project

Public/private repository 🡪 Public ονομάζεται το repository που είναι διαθέσιμο προς όλο τον κόσμο. Μπορεί να δει (όχι να επεξεργαστεί) τον κώδικα μας και το ιστορικό του. Private ονομάζεται το repository που **ΔΕΝ** είναι διαθέσιμο προς όλο τον κόσμο. Αφορά μονάχα τα άτομα που έχει προσκαλέσει ο δημιουργός του.

Organization 🡪 Η ομπρέλα κάτω από την οποία μπορούμε να βάλουμε πολλά repositories σαν ομάδα.

Push 🡪Μεταφόρτωση μιας αλλαγής από τα τοπικά αρχεία μας, στο απομακρυσμένο repository

Pull 🡪 Μεταφόρτωση μιας αλλαγής από το απομακρυσμένο repository ,στα τοπικά μας αρχεία.Αυτο συνήθως συμβαίνει όταν παραπάνω από ένας developer επιρρεάζει αρχεία από το repository. Όλοι οι υπόλοιποι πρέπει να κάνουν pull

Clone 🡪 Μεταφόρτωση ΟΛΟΚΛΗΡΟΥ repository στον υπολογιστή μας

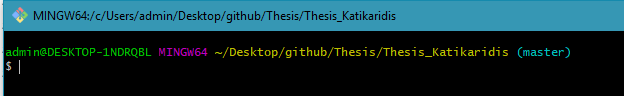
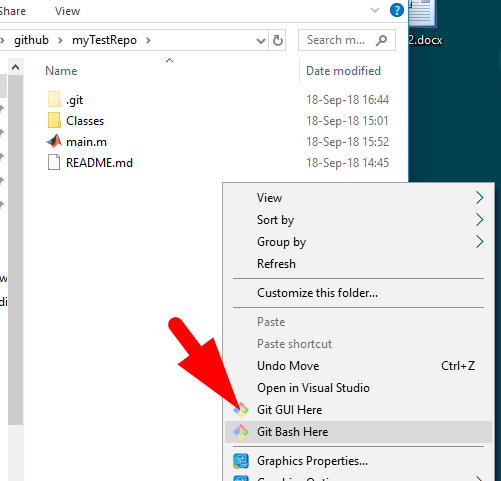
Revert 🡪Αναστροφή ενός commit. Πρακτικά, αναστρέφει το αποτέλεσμα ενός commit

Origin 🡪 Το github application, στο υπόβαθρο, δουλεύει με χρήση ssh. Origin ονομάζεται ο μεσολαβητής της επικοινωνίας του Η/Υ μας υπολογιστή με το απομακρυσμένο repository

Fetch ή sync 🡪 Μεταφόρτωση όλων των αλλαγών από το απομακρυσμένο repository στα τοπικά αρχεία μας.

# Προχωρημένες εντολές και χρήση τερματικού

Σε αντίθεση με το τερματικό του Linux, η εφαρμογή του Windows GitHub έχει περιορισμένες δυνατότητες. Η πλήρης χρήση του github απαιτεί εντολές από το git bash (<https://gitforwindows.org/> )



Παρακάτω αναγράφονται οι εντολές που χρειαζόμαστε

