Εργαλείο Git

Η βασική ιδέα είναι οτι αν θέλω να φτιάξω ενα project, μπορεί να θέλω να αλλάξω κάτι μικρό για να δοκιμάσω κάτι. Αυτό που κάνω συνήθως είναι να αντιγράψω το αρχείο (η τα αρχεία) σε ένα νέο directory (η να το μετονομάσω σε eleanna17.cpp) . Αντί να το κάνω αυτό, χρησιμοποιώ ένα εργαλείο που το κάνει αυτόματα για μένα, το λεγόμενο git.

Συνεπώς ανοιγούμε το terminal και εγκαθιστούμε το git (sudo apt-get install git) .

Κάποια configurations

Για να κάνει την δουλεία του το git χρειάζεται κάποιο όνομα, και το email (δεν είμαι σιγουρος γιατι) οπότε εκτελούμε (μόνο μία φορά απο τη στιγμή που εγκαταστήσαμε το git):

git config --global user.email "you@example.com"
git config --global user.name "Your Name"

Δημιουργία repository

Φτιάχνουμε ενα directory οπου θα ξεκινήσουμε το project, και μέσα σε αυτό εκτελούμε

git init

Προσθήκη αρχείου στο repository

Έστω οτι έχουμε φτιάξει ένα αρχείο, το mpoufos.c . Τότε λέμε

git add mpoufos.c

Για να είμαστε σωστοί θα πρέπει κάθε φορά που βάζουμε ενα αρχείο να το κάνουμε και **commit**, δηλαδή να ενημερώνουμε το repository οτι έγινε μια αλλαγή , ώστε να μπορούμε να επιστρέψουμε στην προηγούμενη κατάσταση άμα αλλάξαμε γνώμη.
Το commit είναι σαν checkpoint, στο οποίο μπορείς να επιστρέφεις αμα θές. Όσα περισσότερα commits τόσο πιο ασφαλής είσαι οτι δεν θα χάσεις κάποια δουλειά σου.

Συνεπώς για να κάνουμε commit εκτελούμε:

git commit -m “Some commenting about the changes I made, for me to remember (or for others if I am not alone) “

Έτσι έχω ενα checkpoint στο οποίο μπορώ να γυρίσω αμα αλλάξω πράγματα και τελικά το μετανιώσω.
Για να δώ τα διαθέσιμα checkpoints μου μπορώ να εκτελέσω

Wandering around

git log ή
git log --online

Το αποτέλεσμα της δεύτερης εντολής θα μοιάζει κάπως έτσι:

```
[marjus@myarch tlproject]$ git log --oneline
35dfae4 (HEAD) Just wrote 42
352c021 (origin/master, origin/Google-maps, master) I just added my first file o
n my local git
```

Όπου το ***head*** είναι το τρέχων checkpoint ενώ οι πρώτοι χαρακτήρες σε κάθε γραμμή είναι το ***hash*** κάθε checkpoint, το οποίο είναι και η αναφορά μας σε κάθε commit.
Συνεπώς έστω οτι αλλάξαμε γνώμη και θέλουμε αν ελέγξουμε κάποιο προηγούμενο commit.
Τότε αρκεί να εκτελέσουμε

git checkout <hash i.e. 352c021>

Τότε στο directory μας θα βλέπουμε τα αρχεία που είχαμε σε αυτό το commit με τα αντίστοιχα περιεχόμενα τους. Μπορούμε να τα αλλάξουμε και να δημιουργήσουμε νέα, καθώς και να τα επεξεργαστούμε καθώς και να κάνουμε commits, ωστόσο για να διατηρηθούν οι αλλαγές πρέπει να δημιουργήσουμε ενα νέο ***branch***, ενώ βρισκόμαστε στο συγκεκριμένο checkpoint (αυτό το βλέπουμε επειδη εκεί είναι το **head.** Το head είναι το σημείο που επεξεργαζόμαστε την εκάστοτε “στιγμή”.

Branches

Και κάπου τώρα ήρθε η ώρα να μιλήσουμε για τα branches. Η ιδέα είναι οτι αν θέλουμε να προσθέσουμε κάποιο καινούριο χαρακτηρηστικό στο project που έχουμε χτίσει μέχρι τώρα, δεν το κάνουμε στο ίδιο directory, αλλά φτιάχνουμε ένα νέο. Στο git δεν υπάρχουν ακριβώς directories αλλά μια πιο σύνθετη ιδέα, τα branches όπου πρακτικά έχουν και το σχετικό ιστορικό με τις αλλαγές που έχεις κάνει μεχρι τώρα χωρίς αυτό να φαίνεται στο βασικό project.

Το βασικό project βρίσκεται στο branch ***master.*** Σε αυτό θεωρητικά θα υπάρχει το ολοκλήρωμένο project, ενώ στα υπόλοιπα , κομμάτια που θα ενωθούν με αυτό. Συνεπώς όταν εγώ φτιάξω ενα νέο branch, θα αντιγράψει τα πάντα που υπάρχουν στο master στο νέο branch και πλέον τα commit history θα αποκτήσουν διαφορετική διακλάδωση. Οτι commit κάνω στο νέο branch δεν φαίνεται στο master και αντιστροφά.

Όταν θα έχω ολοκληρώσει το έργο μου στο branch που έφτιαξα και είμαι σίγουρος οτι θέλω να το εντάξω στο βασικό project, τότε κάνω ***merge*** το σχετικό branch με το master. Αυτό θα το δούμε παρακάτω.

Για να **δημιουργήσουμε** ένα νέο branch εκτελούμε:

git branch <mybranchname> ή

git checkout -b <mybranchname> Έτσι δημιουργώ και “πηγαίνω” στο νέο μου branch

Για να δώ τα branches μου που έχω δημιουργήσει ως τώρα εκτελώ:

git branch

Με αστερίσκο (*) φαίνεται σε ποιό branch βρίσκομαι τώρα.
Για να περιηγηθώ στα branches χρησιμοποιώ την εντολή:

git checkout <branchname i.e. master>

Ανάλογα σε ποιο branch είμαι η εντολή ***git log --oneline*** θα βγάλει διαφορετικά αποτελέσματα, που αφορούν commits του εκάστοτε branch.

```
[marjus@myarch tlproject]$ git log --oneline
dac12fd (HEAD -> master) Establish first things first
974bca2 Revert "17 reborn"
b83aa6b 17 reborn
cfad76b Revert "My child 17"
96c9c6b My child 17
352c021 (origin/master) I just added my first file on my local git
```

Συνεπώς αν εκτελέσω **git checkout 974bca2** θα με οδηγήσει σε ένα προηγούμενο commit του branch master, και αν εκεί δημιουργήσω καινούριο branch (**git checkout -b "evolution"**), π.χ. με όνομα *evolution,* θα περιέχει οτιδήποτε είχα κάνει μέχρι εκείνο το checkpoint.

Αν θέλω να διαγράψω ένα branch (κατα προτίμηση μετά το merging) μπορώ να εκτελέσω:

git branch -d <branchname>

Merge

Με το *merge* ολοκληρώνεται και η ιδέα του git όπου πλέον έχω ολοκληρώσει το έργο μου σε κάποιο branch και θέλω να το εντάξω στο συνολικό project.
Αυτό που θέλω να πετύχω είναι ότι αλλαγές έχουν γίνει στο branch και οτι αλλαγές μπορεί να έχω κάνει στο master να παραμείνουν και να βρεθούν στο ίδιο project, μαζί και με το commit history σε περίπτωση που αλλάξω γνώμη.
Όπως πιθανόν να σκεφτήκατε ήδη αυτό δεν μπορεί να γίνει πάντα, π.χ. αν εγώ αλλάζω το ίδιο αρχείο, με διαφορετικό τρόπο, και στη συνέχεια προσπαθήσω να κάνω merge αυτό δεν μπορεί να συμβεί γιατί ένα απο τα δύο αρχεία πρέπει να διαγραφεί.
Το git δεν θα αποφασίσει για μας ποιό , αντίθετα θα μας τυπώσει ένα μύνημα που θα αναφέρει όλα τα conflicts που υπάρχουν και **θα σταματήσει την διαδικάσία του merging στη "μέση".**
Τότε εμείς θα πρέπει να αντιμετωπίσουμε τα conflicts, και όταν δεν υπάρχουν πλέον conflicts θα κάνουμε **commit** και το git θα συνεχίσει την διαδικασία του merging μόνο του.

Συνεπώς όταν θέλω να κάνω merge πηγαίνω στο master (git checkout master) και εκτελώ:

git merge <branchname>

Αν δεν πετύχει λόγο conflict, δηλαδή υπάρχουν 2 αρχεία με το ίδιο όνομα και διαφορετικό περιεχόμενο, τότε θα δημιουργήσει **ένα** νέο αρχείο με αυτό το όνομα όπου θα περιέχει το περιεχόμενο των δυο αρχείων το **ένα κάτω απο το άλλο ως εξής:**

```
[marjus@myarch tlproject]$ cat first.txt
<<<<<<< HEAD
This is the very first file created
=======
42
>>>>>>> branch42
```

Το αρχείο του master περιείχε το “This is the very first file created” και το branch περιείχε το “42”.
Τώρα για να αντιμετωπίσω το conflict αλλάζω το αρχείο όπως θέλω, και στη συνέχεια εκτελώ ***git add first.txt*** και ***git commit -m “…***” .
Έτσι ολοκληρώνεται το merging και έχει αντιμετωπιστεί το conflict.
Αν δεν υφίσταται conflict το merging θα πετύχει κανονικά. Συνεπώς θα είχε νόημα να διαγράψουμε το branch πιθανώς (***git branch -d branch42***).

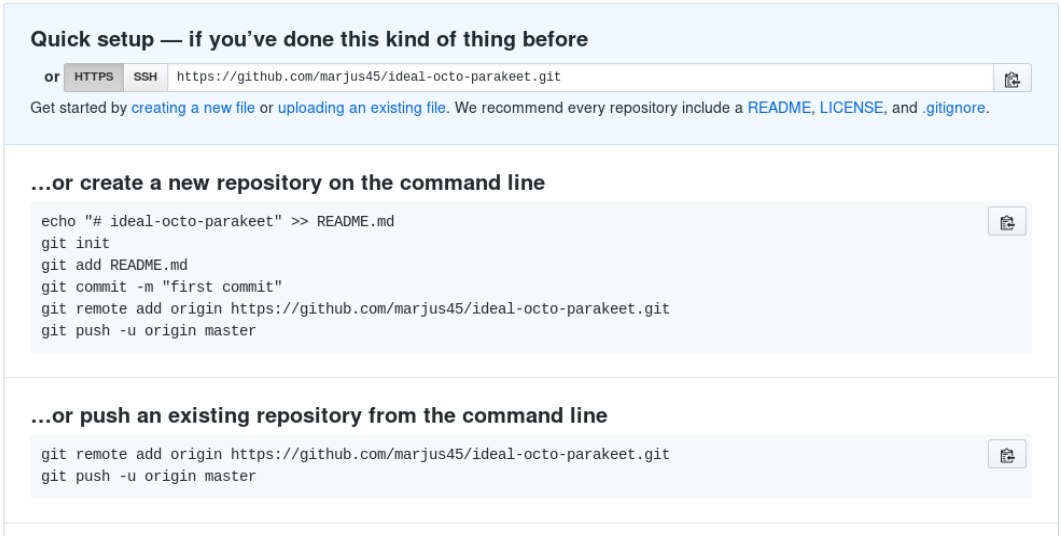
Επειδή είναι ιδιαίτερο το θέμα αμα δεν έγινε πλήρως κατανοητό μπορείτε να ανατρέξεται εδω.

Github?

Έστω τώρα ότι θέλω να ασχοληθώ με ένα ομαδικό project. Τότε θα ήθελα τις αλλαγές που κάνω να τις βλέπουν όλοι, και πιθανώς τις αλλαγές που κάνουν άλλοι να τις βλέπω και εγώ. Για αυτό το λόγο χρειάζομαι κάποιο web εργαλείο που να συνδυάζεται με το git και να κάνει την ίδια δουλειά που θα έκανα τοπικά.
Τέτοια web εργαλεία είναι το github, gitlab και το gitbucket. Προς το παρών έχουμε student account όποτε το github είναι μαι χαρά, ωστόσο το gitlab προσφέρει απεριόριστες δωρεάν δυνατότητες γενικά, χωρίς να έχω καταλάβει αν υπάρχει κάποια ποιοτική διαφορά ανάμεσα στα εργαλεία.

Τώρα εγώ θα ήθελα να μπορώ να συνδέσω το directory μου με το github, ώστε να μπορώ να κάνω οτι αλλαγές κάνω τοπικά, να φαίνονται στο github και κατ’ επέκταση στην ομάδα.

Αφού δημιουργήσω ενα repository στο github (πατήσω *create repository)* θα μου εφανίσει το παρακάτω:



Το github σου δίνει ήδη οδηγίες για το πως να συνδυάσουμε το git και το τοπικό μας project, και εμείς αυτό θα ακουλουθήσουμε.

Σύνδεση με github

Αυτό που θέλουμε είναι να θυμάτε "για πάντα" οτι αυτό το directory – project που φτιάξαμε τοπικά συνδέεται με το repository του github που μόλις δημιουργήσαμε (ideal-octo-parakeet στην περίπτωση μου)

Θα αντιγράψουμε το link που δείχνει δίπλα απο το **https** | ssh και θα εκτελέσουμε:

```
git remote add origin https://github.com/marjus45/ideal-octo-parakeet.git
```

Το παραπάνω εκτελείτε μία μόνο φορά.
Τώρα έχουμε κάνει την σύνδεση και μπορούμε να "ανεβάσουμε" το branch μας. Έστω οτι είμαστε στο branch master (**git checkout master**).

Επειδή είναι η πρώτη φορά για το συγκεκριμένο branch θα πρέπει να εκτελέσουμε:

git push -u origin master (ή branchname)

Απο δω και πέρα όταν αλλάζουμε κατάσταση (commit) αρκεί να εκτελούμε **git push** ενω βρισκόμαστε στο εκάστοτε directory.
Δεν χρειάζεται να κάνουμε συνέχεια git push (σε αντίθεση με τα commits) , αλλά όποτε θέλουμε να το δούν και οι υπόλοιποι.

Παρένθεση

Επειδή εδώ θα ζητήσει κωδικό, και πιθανόν να ζητάει κωδικό κάθε φορά που θα εκτελείτε git push υπάρχουν κάποιες λύσεις.
Μια κακή λυση είναι να cachareis τον κωδικό για κάποια ώρα π.χ.

git config --global credential.helper 'cache –timeout=3600'

Έτσι του λες οτι για 3600 δευτερόλεπτα θα μου κρατήσεις τον κωδικό. Έτσι όταν στον ζητήσει την πρώτη φορά θα κάνει 1 ώρα μέχρι να στο ξαναζητήσει.

Μια άλλη κακή λύση είναι να “αποθηκεύσεις” τον κωδικό unencrypted ως εξής:

git config credential.helper store

Μια καλή πρακτική είναι ωστόσο να φτιάξεις έναν ssh κωδικό και να το συνδέσεις. Παίρνει λιγότερο απο ένα λεπτό και αξίζει τον κόπο.

Αρχικά ελέγξτε αν έχετε κάποιο ssh κλειδί.

cat ~/.ssh

Αν δεν έχετε δημιουργήστε ως εξής:

ssh-keygen -t rsa -b 4096 -C "*your_email@example.com*"

Πατάτε enter σε ότι σας βγάζει.
Τώρα δημιουργήσατε ένα ιδιωτικό κλειδί και ένα δημόσιο.
Εκτελέστε (είτε είχατε είτε μόλις φτιάξατε κλειδί):

eval "$(ssh-agent -s)"

ssh-add ~/.ssh/id_rsa

Και τώρα αρκεί να αντιγράψουμε το public key που βρίσκεται στο ~/.ssh.
Ένας μη γραφικός τρόπος είναι :

xclip -sel clip < ~/.ssh/id_rsa.pub

Και τώρα αρκεί να το κάνουμε paste στο github. Πηγαίνουμε **settings → SSH ang GPG keys →paste στο σχετικό πλαίσιο → New SSH key .**

Για να το χρησιμοποιήσουμε πλέον εκεί που δημιουργούμε το νέο repository στο github και αναγράφει https | **ssh** , θα επιλέξουμε το ssh και θα κάνουμε αυτό copy paste για να κάνουμε

git remote add origin (ssh link)

Κλείνει Παρένθεση

Αν κάποιος έχει προσθέσει αρχεία σε κάποιο branch στο github η έκανε κάποιο merge με pull request τότε μπορώ (και πρέπει) να τραβήξω τις αλλαγές:

git pull origin <branchname>

Αν έχω κάνει κάποιο merge τοπικά τότε αρκεί να κάνω **git push master** για να ενημερωθεί και το github.

Αν θέλω να διαγράψω ενα branch στο github (πιθανώς γιατί έκανα merge τοπικά, και μετα το διέγραψα τοπικά) μπορώ να κάνω:

git branch --delete origin <branchname>

Pull Request

Θα θέλαμε όταν γίνεται κάποια αλλαγή (merge) να την μαθαίνουν και να ερωτούνται όλοι γιατί ο Κορ μάλλον έχει κάνει μαλακία. Οπότε σε ένα ομαδικό project αρκεί κάποιος να πατήσει το Pull Request στο github το οποίο θα σε ρωτήσει τι με τι θα κάνεις merge και άμα δεν υπάρχουν conflicts τότε σε όλους θα εμφανιστεί (ακόμη και αν είσαι ο μόνος στο project) ένα πράσινο (ή γκρί αν υπάρχουν conflicts) κουμπί merge όπου θα πρέπει όλοι να πατήσουν αν θέλουμε να γίνει το merge. Αν υπάρχουν conflicts τότε θα πρέπει να επιλυθούν για να ολοκληρωθεί το merge όπως και τοπικά. Τότε θα θέλαμε να κάνουμε git push master τοπικά για να ενημερωθούμε και μάλλον να διαγράψουμε τα αντιστοιχα branches.

Τέλος αν θέλουμε να πάρουμε όλο το project σε tar αρχείο (έτσι λειτουργούν γενικά στα πακέτα linux όσοι τα εκγαθιστούν χειροκίνητα) μπορούμε να κάνουμε

git clone link_to_github_project