Σεμινάριο Git & GitLab

Τάκης Παναγόπουλος
Νικόλας Μπομπέτσης

Ανάπτυξη Λογισμικού Τηλεπικοινωνιών και δικτύων 2015 - 2016

Ποιοι είμαστε

- Τάκης Παναγόπουλος
  - Research Associate at ScanLab
    - takisp [at] di [dot] uoa [dot] gr
    - http://scan.di.uoa.gr/mr-takis-panagopoulos/
- Νικόλας Μπομπέτσης
  - Research Associate at ScanLab
    - nbompetsis [at] di [dot] uoa [dot] gr
    - http://scan.di.uoa.gr/mr-nikolas-bompetsis/

Περιεχόμενα

- Τι είναι το git
- Βασική χρήση git
- Δουλεύοντας τοπικά με git
- Δουλεύοντας απομακρυσμένα με git
- Συνεργασία μέσω gitlab
- Workflows και συνεργατικές τεχνικές

Το πρόβλημα ανάμεσα σε ομαδικές εργασίες

- Ένα νέο κομμάτι κώδικα μερικές φορές είναι buggy και θέλουμε
  - Να το αλλάξουμε
  - Να γυρίσουμε πίσω σε σωστή κατάσταση
  - Να δούμε ποιος ανέβασε το λάθος κομμάτι κώδικα
- Δουλεύουμε πολλοί ταυτόχρονα στον ίδιο κώδικα
- Διαγράφουμε κώδικα που μπορεί να χρειαστεί ξανά
- Χρειαζόμαστε back-ups για τη δουλειά μας

Πώς λύνουμε αυτά τα προβλήματα;

- Πώς κρατάμε πολλές εκδόσεις ενός αρχείου;
- Πώς επιστρέφουμε σε μία παλιά έκδοση;

- Η απάντηση σε όλα τα προβλήματα μας!!!!!!!!

Version control

- Μπορούμε να...
  - κρατάμε εκδόσεις στα αρχεία
  - κάνουμε undo αλλαγές
  - συνεργαζόμαστε με άλλους
  - κρατάμε backups των αρχείων μας
  - μοιραζόμαστε εύκολα τον κώδικα με την ομάδα
  - ξέρουμε ποια είναι η «τελευταία» έκδοση

Ιστορία του version control

- CVS – 1990
  - Από τα πρώτα πλήρη version control systems
- Subversion (SVN) – 2000 (ότι χειρότερο υπάρχει)
  - Διορθωμένο CVS για project-wide management
- git – 2005
  - Distributed version control system
- GitHub – 2008 (open source communities)
  - Συνεργατικό περιβάλλον version control
- GitLab – 2013 (Self hosted Git repository manager)

Τι είναι το git?

- Πρόγραμμα που τρέχεις στον υπολογιστή σου
- Εργαλείο από command line
- Ελέγχεις τον κώδικα σου με αυτό

Εγκατάσταση του git

- Linux (Debian, Ubuntu)
  - `apt-get install git`
- Fedora
  - `yum install git`
- Mac
  - `http://git-scm.com/download/mac`
- Windows
  - Δεν μας χρειάζεται αυτό το λειτουργικό!!

Βιβλιογραφία

- git-οβιβλίο: https://git-scm.com/book
- https://try.github.io/
- https://pcottle.github.io/learnGitBranching/

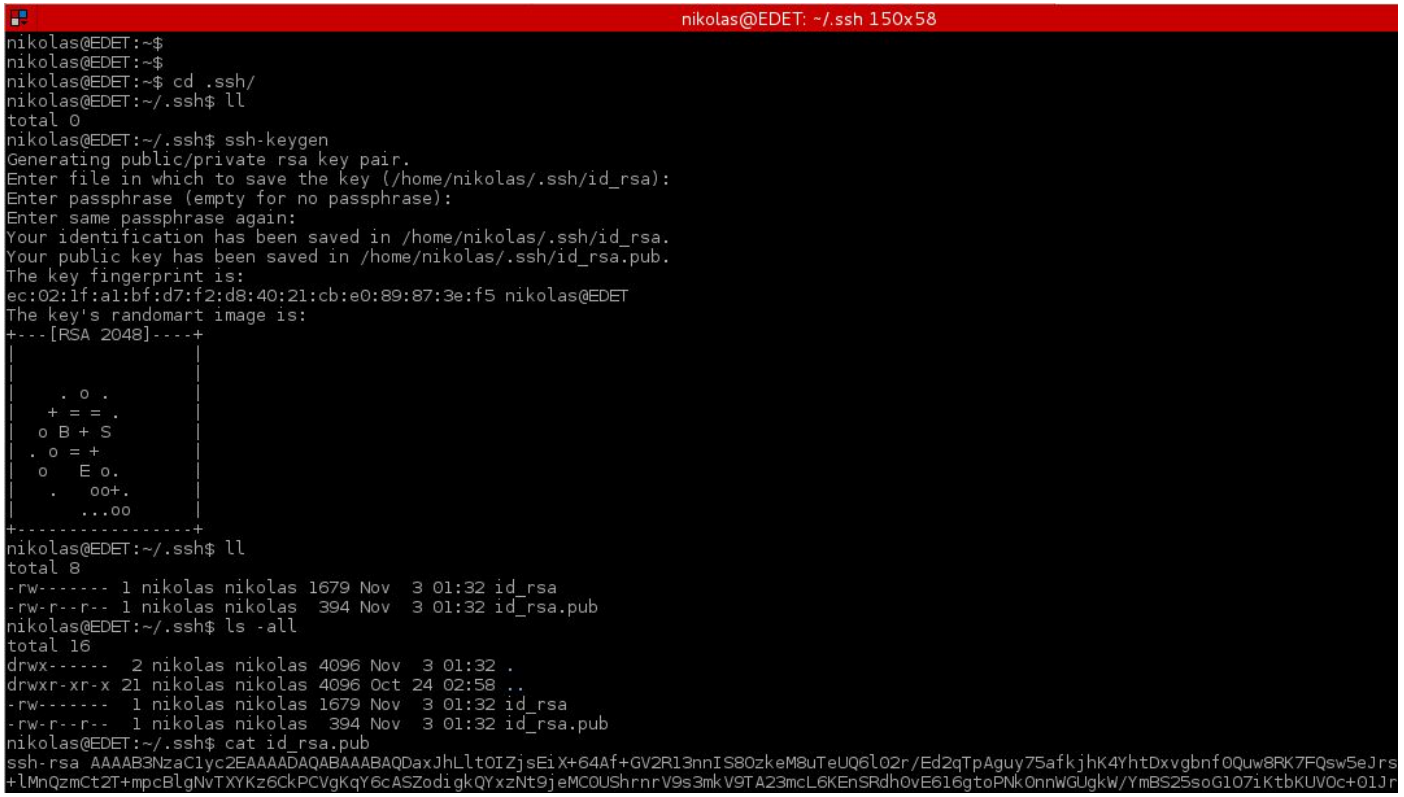
Ρύθμιση του Git

- Πρέπει να πεις στο git ποιος είσαι
  - Το όνομα θα φαίνεται στον κώδικά σου
- git config --global
  - Αλλάζει τις ρυθμίσεις του git
- Πρέπει να πεις στο git ποιος είσαι

```
Nikolass-MacBook-Pro:~nikolas in ~/git_test $ git config --global user.name "Nikolas Bompetsis"
Nikolass-MacBook-Pro:~nikolas in ~/git_test $ git config --global user.email "nbompetsis@di.uoa.gr"
Nikolass-MacBook-Pro:~nikolas in ~/git_test $
```

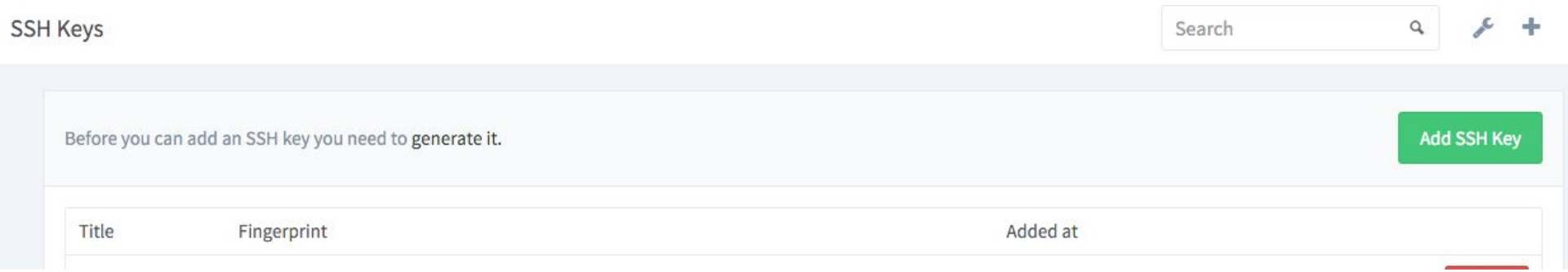
Δημιουργία των ssh -keys

- Ένας τρόπος για να συνδέεσαι στο git



Add new ssh -keys GitLab

- Προσθήκη κλειδιών στο GitLab

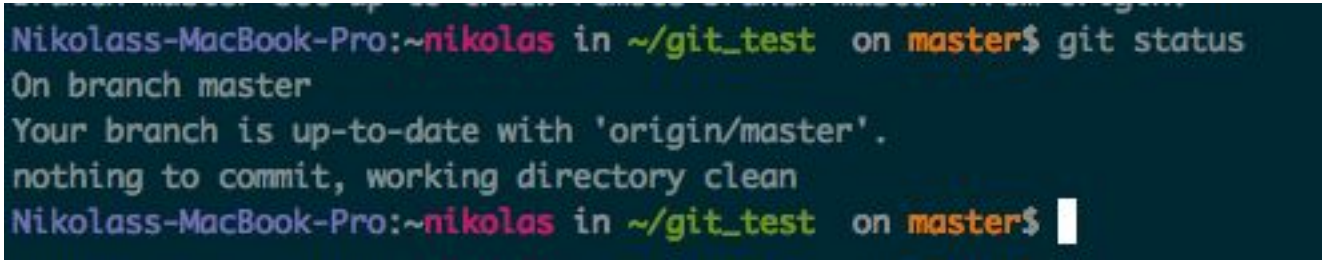


- Πρέπει να βρεθείς στον φάκελο που υπάρχει ο πηγαίος κώδικας του project
- Με αυτές τις εντολές δημιουργείς ένα git repository

```
Nikolass-MacBook-Pro:~nikolas in ~/git_test $
Nikolass-MacBook-Pro:~nikolas in ~/git_test $
Nikolass-MacBook-Pro:~nikolas in ~/git_test $
Nikolass-MacBook-Pro:~nikolas in ~/git_test $ git init
Reinitialized existing Git repository in /Users/nikolas/git_test/.git/
Nikolass-MacBook-Pro:~nikolas in ~/git_test $ touch README.md
Nikolass-MacBook-Pro:~nikolas in ~/git_test $ git add README.md
Nikolass-MacBook-Pro:~nikolas in ~/git_test $ git commit -m "add README"
[master (root-commit) 360a64c] add README
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on master$ git remote add origin git@git.scanlab.gr:team_anap/git-gitlab-testing.git
fatal: remote origin already exists.
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on master$ git push -u origin master
Warning: the ECDSA host key for 'git.scanlab.gr' differs from the key for the IP address '195.134.65.251'
Offending key for IP in /Users/nikolas/.ssh/known_hosts:6
Matching host key in /Users/nikolas/.ssh/known_hosts:10
Are you sure you want to continue connecting (yes/no)? yes
Counting objects: 3, done.
Writing objects: 100% (3/3), 220 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@git.scanlab.gr:team_anap/git-gitlab-testing.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on master$
```

git status

Δείχνει την κατάσταση του κόσμου σου
Εκτέλεσαι την εντολή για να ξέρεις την κατάσταση του τοπικού σου repo



git add <file>

Αυτή η εντολή ζητάει από το git να προσθέσει ένα νέο αρχείο στο τοπικό repository

Πρέπει να ακολουθείται από ‘commit’

Παίρνει ως παράμετρο το νέο αρχείο

ή ένα γενικό μοτίβο

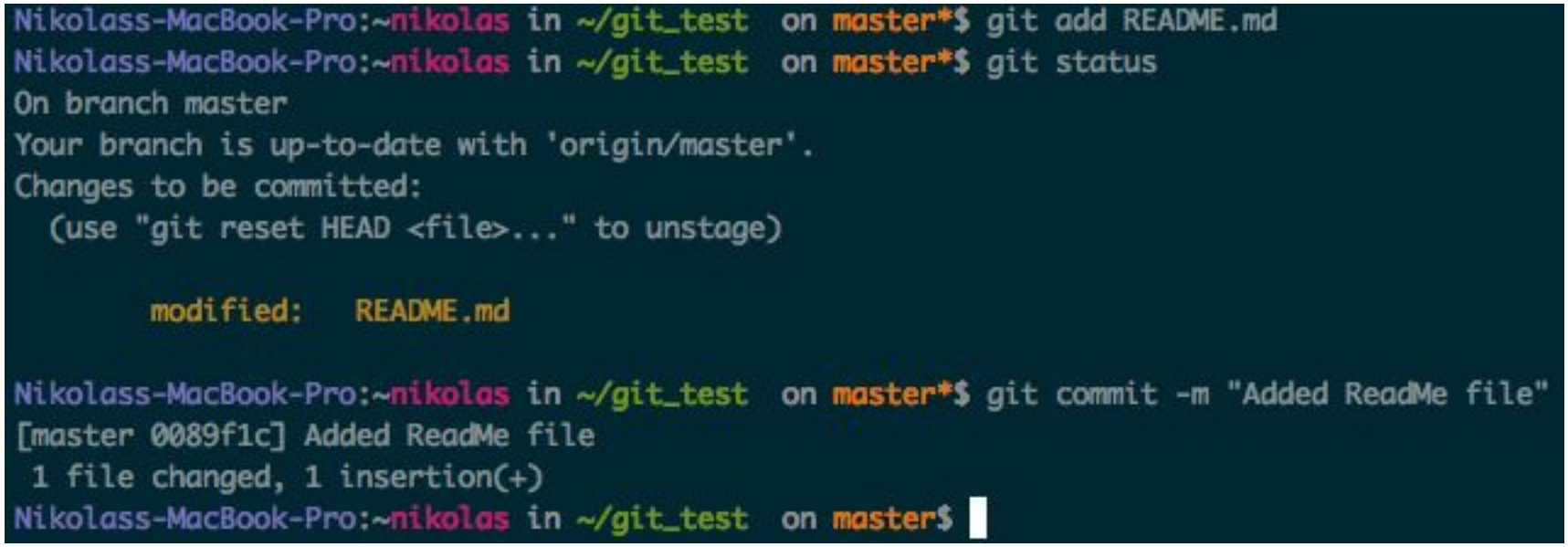
. όλα τα αρχεία

'*.txt' όλα τα αρχεία .txt

```
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on master$ vi README.md
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on master*$ cat README.md
Hello World of git!
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on master*$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on master*$
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on master*$ git add README.md
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on master*$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   README.md

Nikolass-MacBook-Pro:~nikolas in ~/git_test  on master*$
```

Τι είναι ένα «commit αντικείμενο»?

- Ένα στιγμιότυπο του κόσμου
  - Το σύνολο όλων των αρχείων και φακέλων του project μας
  - Με τα περιεχόμενά τους σε μία στιγμή του χρόνου
- Περιλαμβάνει ένα περιγραφικό μήνυμα
- Καταγράφει μετα-δεδομένα
  - Ημερομηνία
  - Δημιουργό
- Έχει ένα μοναδικό αναγνωριστικό
  - π.χ. 5e0dc079899ef4b13f9fa78a53952310f94


Nikolass-MacBook-Pro:~nikolas in ~/git_test on master*$ git add README.md
Nikolass-MacBook-Pro:~nikolas in ~/git_test on master*$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)
modified: README.md
Nikolass-MacBook-Pro:~nikolas in ~/git_test on master*$ git commit -m "Added ReadMe file"
[master 0089f1c] Added ReadMe file
1 file changed, 1 insertion(+)
Nikolass-MacBook-Pro:~nikolas in ~/git_test on master$

git help command

- Βοήθεια σχετικά με κάποιο git command
- `git help add`
- `git help commit`

Βασικό git workflow

- `vim README`
- `git add README`
- `git status`
- `git commit -m "Changed README"`

.gitignore

- Αρχείο στον κεντρικό φάκελο του repo
- Μέσα γράφεις μία λίστα αρχείων
  - Ένα αρχείο ανά γραμμή
- Τέτοιου είδους αρχεία αγνοούνται από το git
  - Δεν προστίθεται με `git add .`
  - Δεν φαίνονται στο `git status`
- Μπορεί να περιέχει μοτίβα αρχείων
  - `*.swp`

Παράδειγμα `.gitignore`

```
node_modules/
bower_components/
*.pyc
*.swp
*.log
config/local.json
nohup.out
client/dist/
API/venv/
```

`git rm`

- Διαγράφει ένα αρχείο και ενημερώνει το git
  - Πρέπει να ακολουθείται από commit
- Κατά μία έννοια το «αντίθετο» του add

`git mv`

- Μεταφέρει ένα αρχείο και ενημερώνει το git
  - Πρέπει να ακολουθείται από commit
- `git mv`:
  - mv <old path> <new path>
  - git rm <old path>
  - git add <new path>

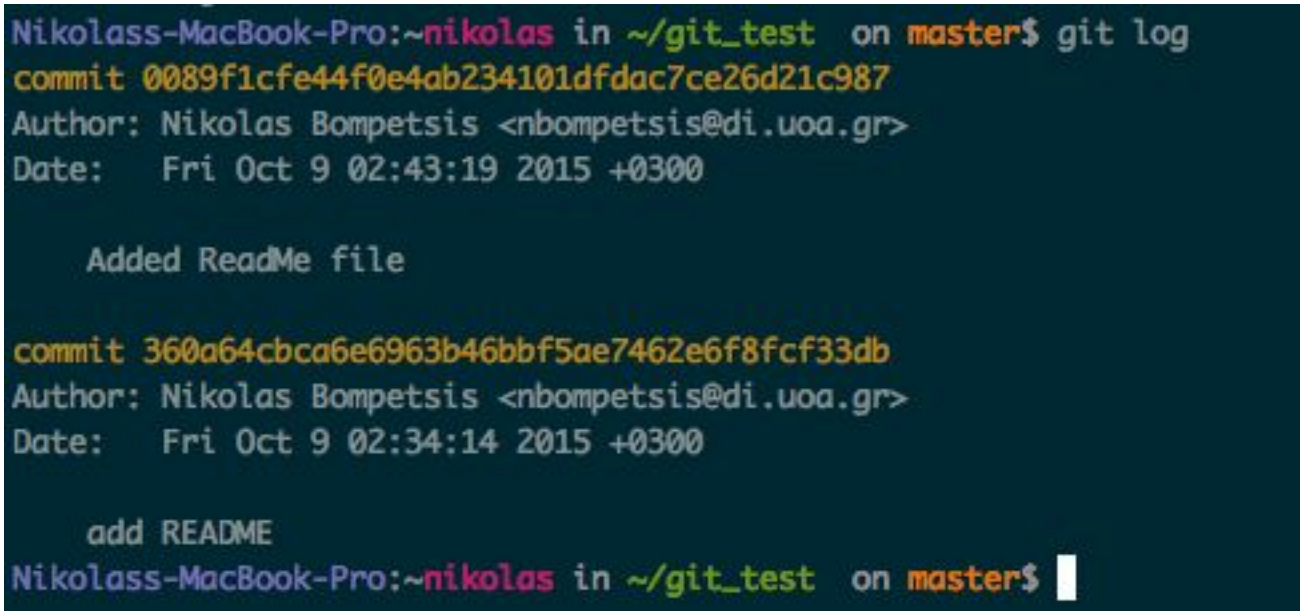
Ιστορικό

- Ο κώδικας αποτελείται από μία σειρά από commits
- Τα commits βρίσκονται σε χρονολογική σειρά

Git log

- Δείχνει το ιστορικό
- Με το log βλέπουμε
  - Λίστα με τα Commits
  - Ποιος έκανε το κάθε commit
  - Περιγραφή
  - Χρονική σειρά

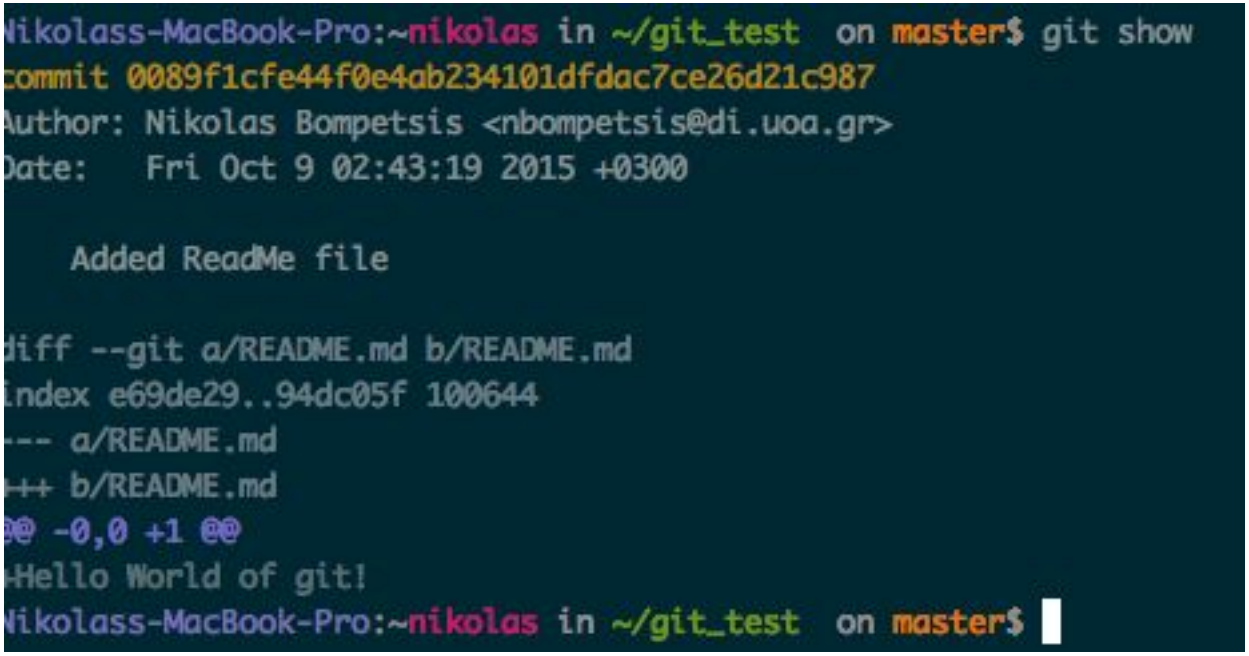
Git log


Nikolass-MacBook-Pro:~nikolas in ~/git_test on master$ git log
commit 0089f1cfe44f0e4ab234101dfdac7ce26d21c987
Author: Nikolas Bompetsis <nbompetsis@di.uoa.gr>
Date: Fri Oct 9 02:43:19 2015 +0300
Added ReadMe file
commit 360a64cbca6e6963b46bbf5ae7462e6f8fcf33db
Author: Nikolas Bompetsis <nbompetsis@di.uoa.gr>
Date: Fri Oct 9 02:34:14 2015 +0300
add README
Nikolass-MacBook-Pro:~nikolas in ~/git_test on master$

Git show

- Δείχνει τι έκανε ένα συγκεκριμένο commit
- Τρέχει με παράμετρο ένα αναγνωριστικό
  - Ή ένα μοναδικό πρόθεμα αναγνωριστικού
- Με το show βλέπουμε:
  - Όλα όσα βλέπαμε με το git log
  - Όλα όσα άλλαξε ένα συγκεκριμένο commit

Git show


ikolass-MacBook-Pro:~nikolas in ~/git_test on master$ git show
commit 0089f1cfe44f0e4ab234101dfdac7ce26d21c987
Author: Nikolas Bompetsis <nbompetsis@di.uoa.gr>
Date: Fri Oct 9 02:43:19 2015 +0300
Added ReadMe file
diff --git a/README.md b/README.md
index e69de29..94dc05f 100644
--- a/README.md
+++ b/README.md
@@ -0,0 +1 @@
+Hello World of git!
ikolass-MacBook-Pro:~nikolas in ~/git_test on master$

Αναγνωριστικά των commits

- Hash του περιεχομένου και των μεταδεδομένων του commit
- Μοναδικό για κάθε commit
  - Διαφορετικό για κάθε commit ακόμη και μεταξύ διαφορετικών repositories!
- π.χ. 8436b9f2457b55b1c81edf50d03ff48283
- Μπορούμε να αναφερθούμε και με ένα πρόθεμα (τουλάχιστον 4 χαρακτήρες)
  - 8436b
  - Αρκεί να είναι μοναδικό μέσα στο repo

`git diff`

- Δείχνει τι άλλαξε στο φάκελό μας για το οποίο το git δεν έχει ενημερωθεί ακόμα
  - Μας λέει τι θα γίνει add αν τρέξουμε git add
- Μπορούμε να το τρέξουμε μόνο του
- Ή να του δώσουμε παράμετρο συγκεκριμένα αρχεία που μας ενδιαφέρουν
- Το output μοιάζει με το `git show`

`git diff --staged`

- Δείχνει τι αλλαγές έχουν γίνει που θα καταγραφούν στο επόμενο commit
  - Μας λέει τι έχει γίνει ήδη git add

`git diff HEAD`

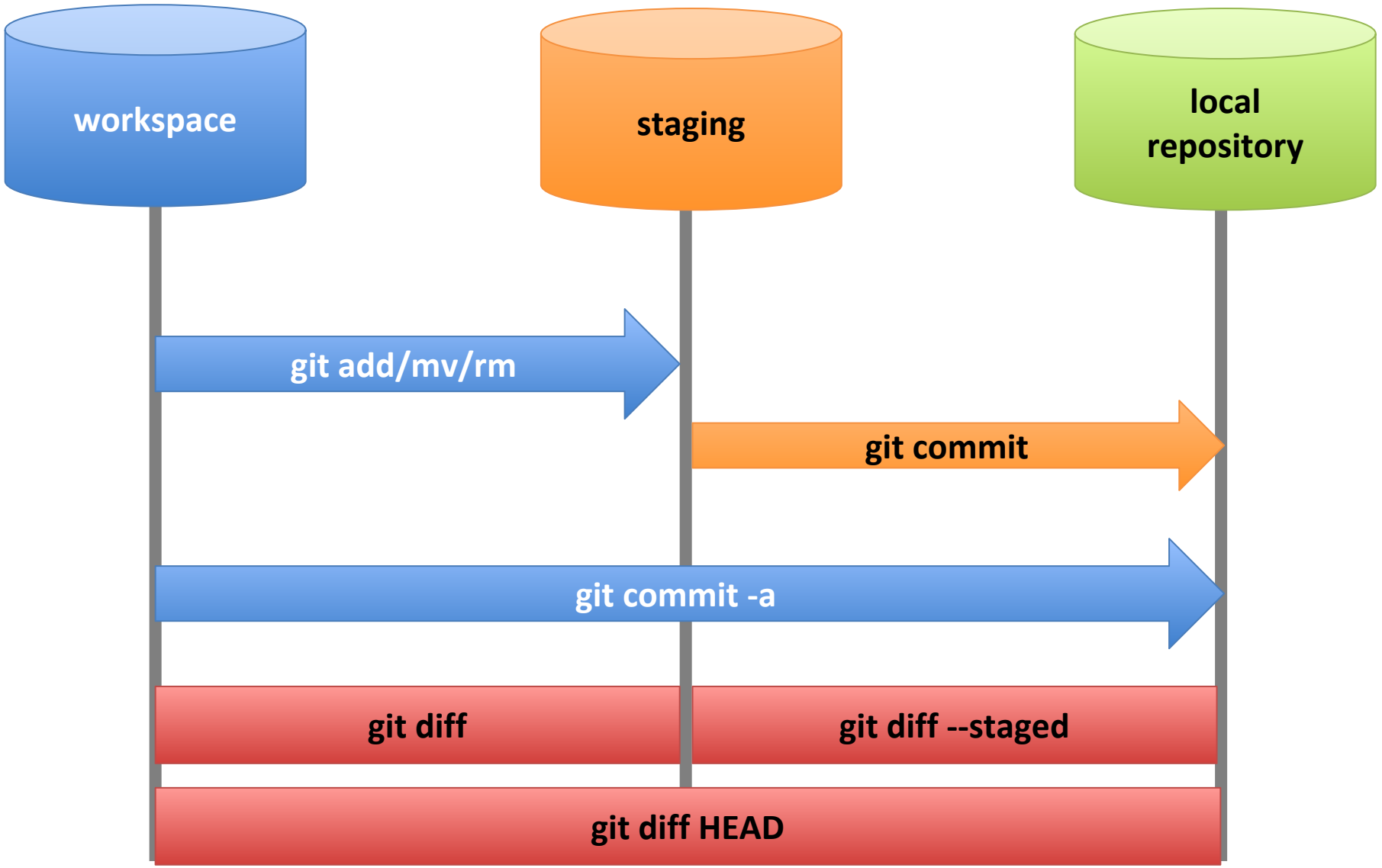
- Δείχνει τι αλλαγές έχουν γίνει στο σύστημα από το τελευταίο commit

Git tag

```
Nikolass-MacBook-Pro:~nikolas in ~/Desktop/AgentPython  on master$ git tag -a v1.4 -m 'my version 1.4'
Nikolass-MacBook-Pro:~nikolas in ~/Desktop/AgentPython  on master$ git tag show v1.4
Nikolass-MacBook-Pro:~nikolas in ~/Desktop/AgentPython  on master$ git show v1.4
tag v1.4
Tagger: Nikolas Bompetsis <nbompetsis@di.uoa.gr>
Date:   Tue Nov 3 02:36:39 2015 +0200

my version 1.4

commit 6d0265b1557b7e7f2cd4b7a6a041a046a5e5f46f
Author: Nikolas Bompetsis <nbompetsis@di.uoa.gr>
Date:   Tue Oct 27 22:04:40 2015 +0200

    Added the else statement to close the file

diff --git a/file/file_parser_reader.py b/file/file_parser_reader.py
index e4f3ef4..1aa889e 100644
--- a/file/file_parser_reader.py
+++ b/file/file_parser_reader.py
@@ -26,7 +26,9 @@ class FileParserReader:
             print("Could not convert data to an integer.")
         except:
             print("Unexpected error:", sys.exc_info()[0])
-
+        else:
+            print arg, 'has', len(self.file.readlines()), 'lines'
+            self.file.close()

     def getListOfJobs(self):
         jobsList = []
Nikolass-MacBook-Pro:~nikolas in ~/Desktop/AgentPython  on master$
```

Staging area

- Ο εικονικός «χώρος» στον οποίο μπαίνουν οι αλλαγές μας όταν κάνουμε git add
- Μας επιτρέπει να προετοιμάσουμε ένα commit

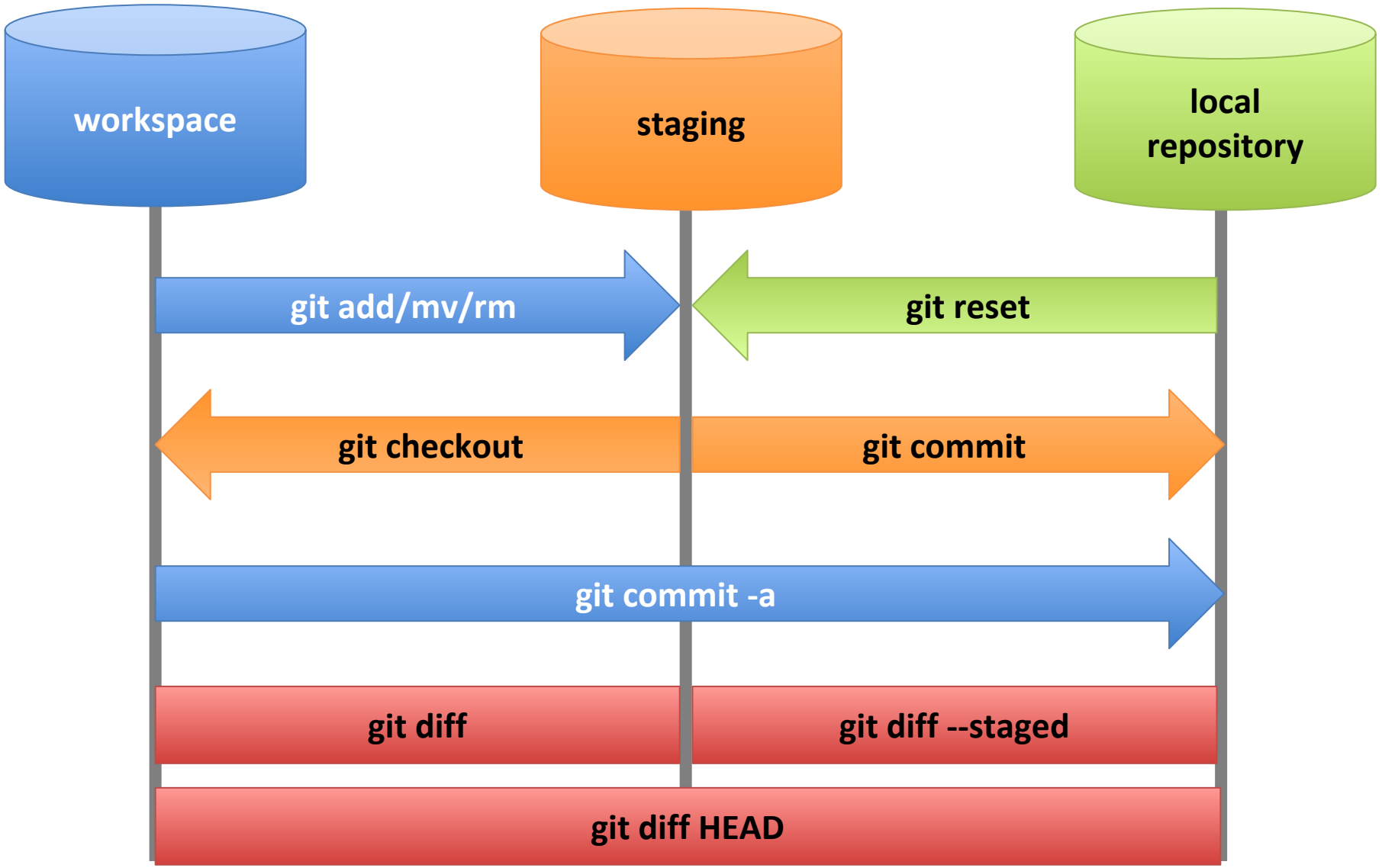

workspace
staging
local repository
git add/mv/rm
git commit
git commit -a
git diff
git diff --staged
git diff HEAD

git checkout <file>

- Ακυρώνει τις αλλαγές μας σε ένα αρχείο
  - που δεν έχουν γίνει stage
  - αντιγράφοντας την κατάσταση του staging area (συνήθως ίδια με το τελευταίο commit) στο working copy

`git reset`

- Αφαιρεί πράγματα που έχουν μπει στο staging area
  - αντιγράφοντάς τα από το πιο πρόσφατο commit
- Αναιρεί μία ενέργεια `git add`


workspace
staging
local repository
git add/mv/rm
git reset
git checkout
git commit
git commit -a
git diff
git diff --staged
git diff HEAD

Branches

- Η βασικότερη λειτουργία του git
- Επιτρέπει να διατηρούμε διαφορετικές εκδόσεις του κώδικά μας
  - Stable
  - Unstable
- Κάθε branch
  - Έχει ένα **όνομα**
- Περιέχει διαφορετικό ιστορικό, με διαφορετικά commits
- Ενδεχομένως κάποια commits να είναι κοινά ανάμεσα σε branches

git branch

`git branch <name>`

- Δημιουργεί ένα νέο branch
- Του δίνουμε ως παράμετρο το όνομα του νέου branch που θέλουμε
- Το νέο branch είναι πανομοιότυπο με το υπάρχον τρέχον (περιέχουν τα ίδια commits)

`git branch`

- Δείχνει τι branches υπάρχουν
- Το τρέχον branch σημειώνεται με *

git checkout <branch>

- Αλλάζει το τρέχον branch
- Το τρέχον branch στον κόσμο του git αναφέρεται με το όνομα “HEAD”
- Το git checkout θέτει το HEAD στο branch που δίνεται ως παράμετρος

`git branch -d <branch>`

- Διαγράφει το branch που περνάς ως παράμετρο
- Τα commits δεν διαγράφονται, μόνο το branch που δείχνει σε αυτά

master

- Το προεπιλεγμένο branch όταν δημιουργούμε ένα νέο repository (με `git init`)
- Στα περισσότερα projects, το branch που περιέχει τον ‘τρέχοντα’ κώδικα
- Συνήθως φτιάχνουμε νέο branch ως αντίγραφο του master

master

```
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on master$ git branch
* master
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on master$ git branch new_feature
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on master$ git branch
* master
  new_feature
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on master$ git checkout new_feature
Switched to branch 'new_feature'
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on new_feature$ git branch
  master
* new_feature
Nikolass-MacBook-Pro:~nikolas in ~/git_test  on new_feature$
```

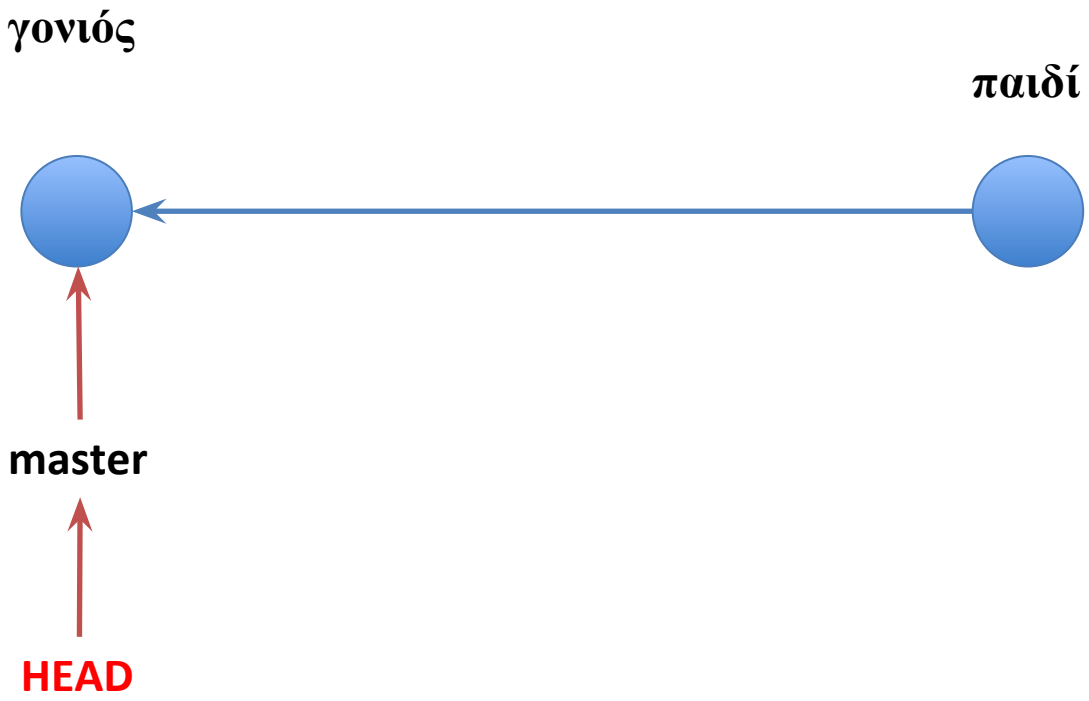
`git checkout -b <branch>`

- Δημιουργεί νέο branch και αλλάζει το τρέχον branch σε <branch>
- git checkout -b <branch> σημαίνει:
  - git branch <branch>
  - git checkout <branch>

Ο γράφος του git

- Το git είναι ένα **σύστημα επεξεργασίας γράφων**
- **Κάθε commit είναι ένας κόμβος**
- Κάθε commit έχει **γονιό** το προηγούμενο commit του
- Ένα branch δείχνει σε ένα commit
- Το HEAD δείχνει στο τρέχον branch
- Ένα branch επιτρέπει τη δημιουργία **διακλαδώσεων** στο γράφο

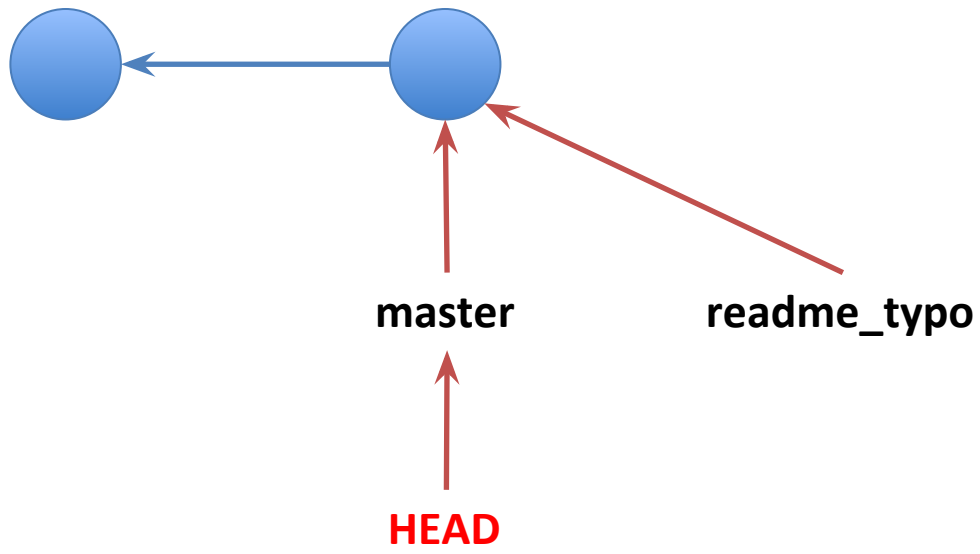
git commit



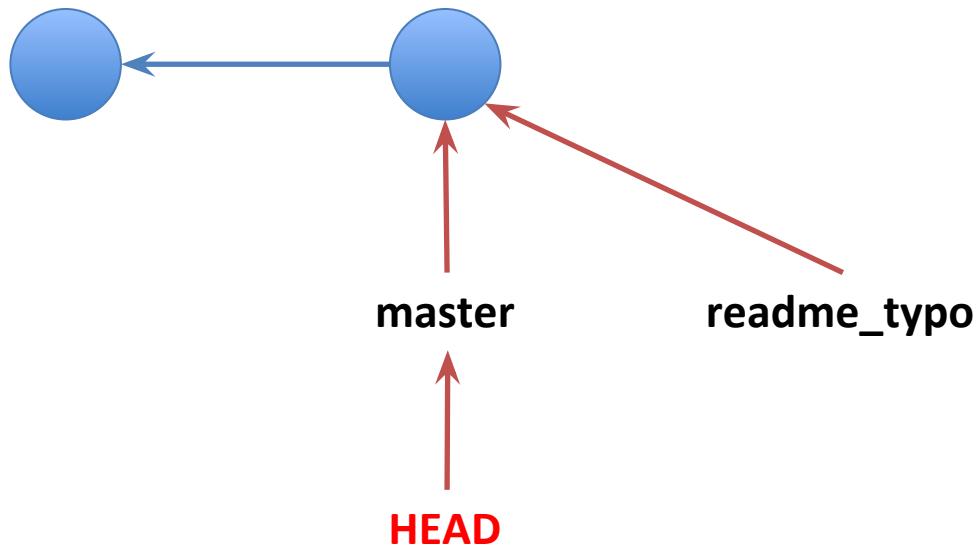
git commit

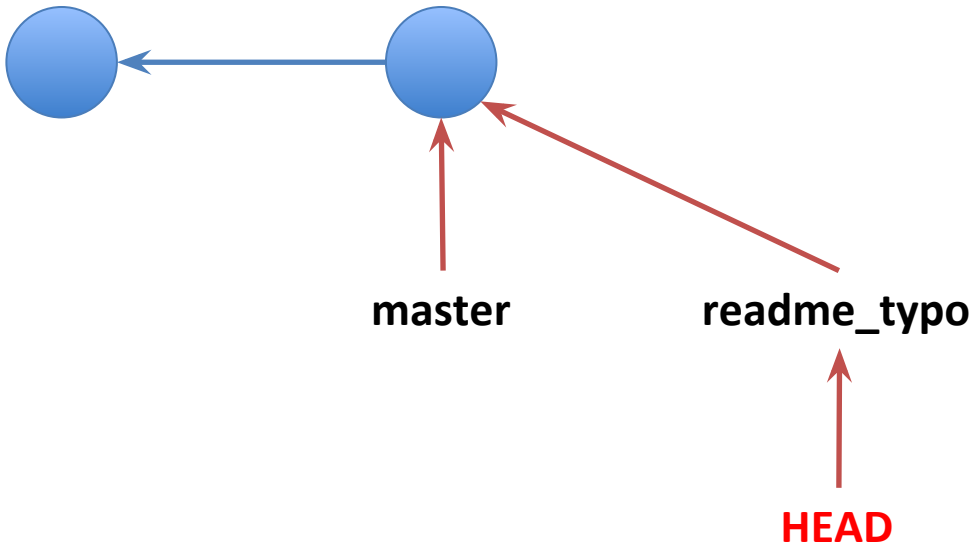
- Τώρα μπορούμε να μιλήσουμε για commits με όρους γράφων
- git commit
  - Δημιουργεί ένα νέο commit αντικείμενο
  - Ορίζει τον **γονιό** του να είναι το commit αντικείμενο στο οποίο δείχνει το τρέχον branch (δηλαδή το branch στο οποίο δείχνει HEAD)
  - Μεταφέρει το τρέχον branch στο νέο commit

git branch readme_typo

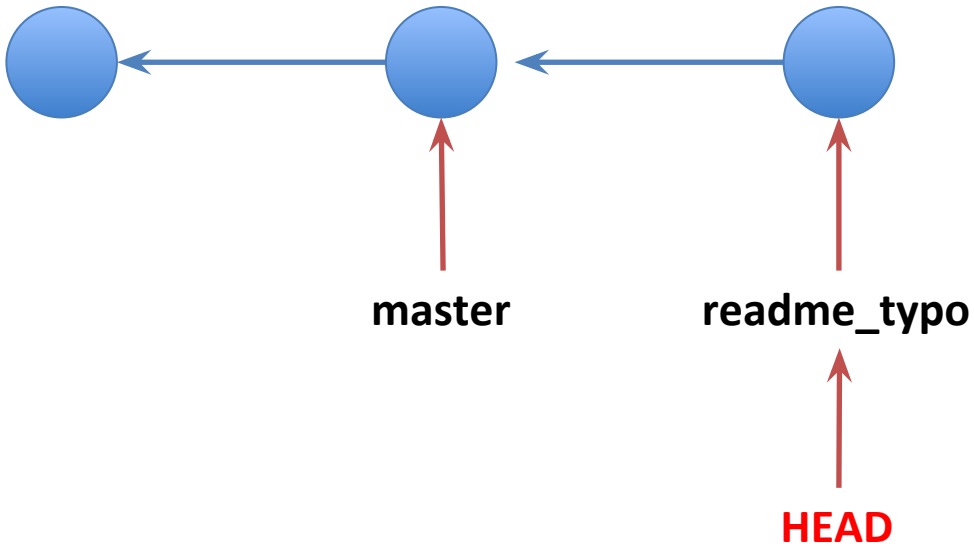


git checkout readme_typo

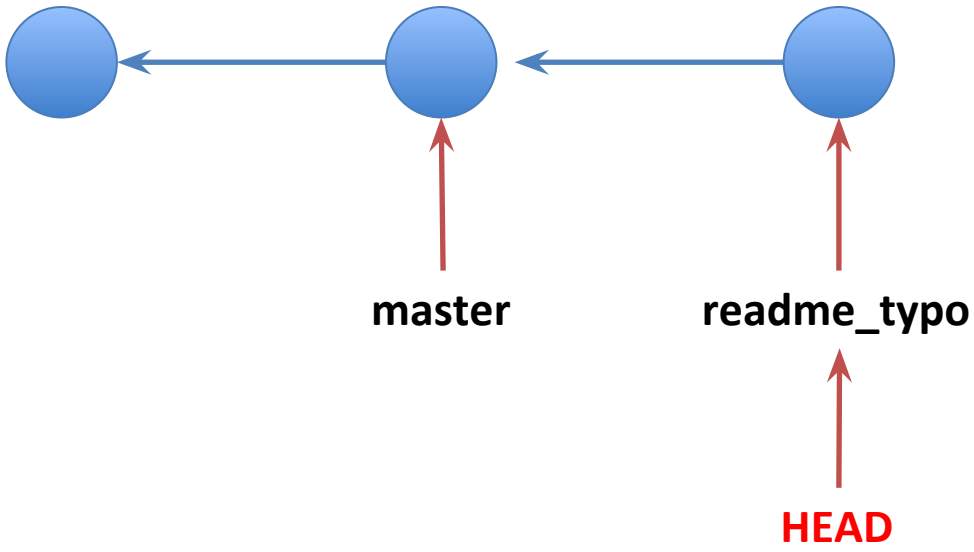



master
readme_typo
HEAD

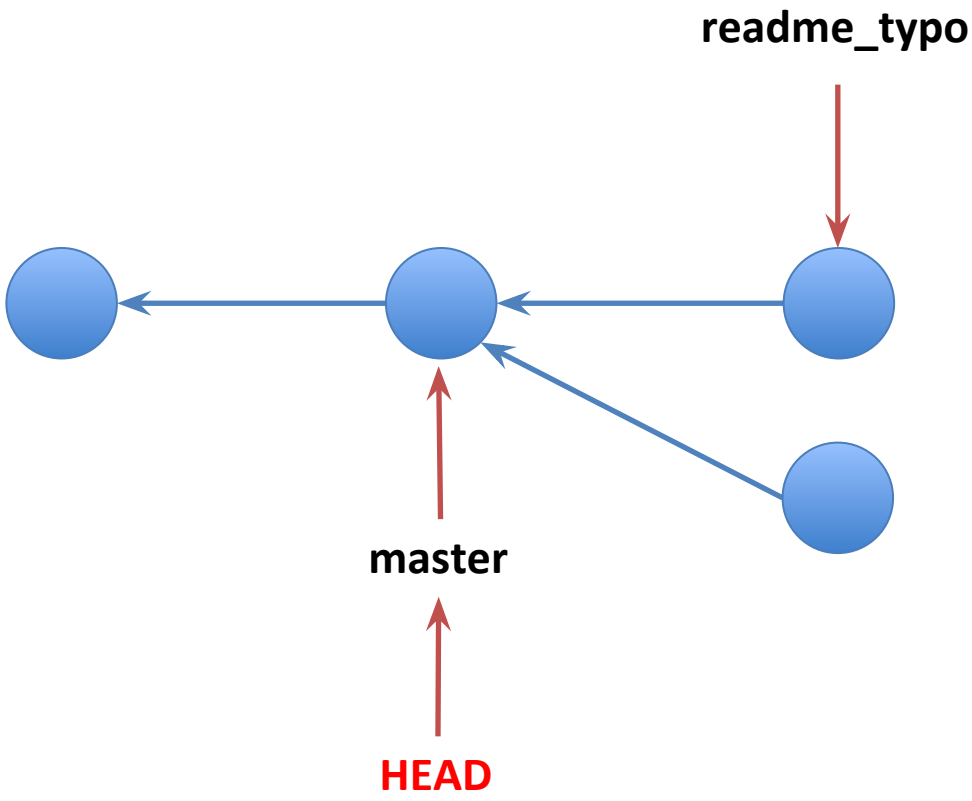
git commit



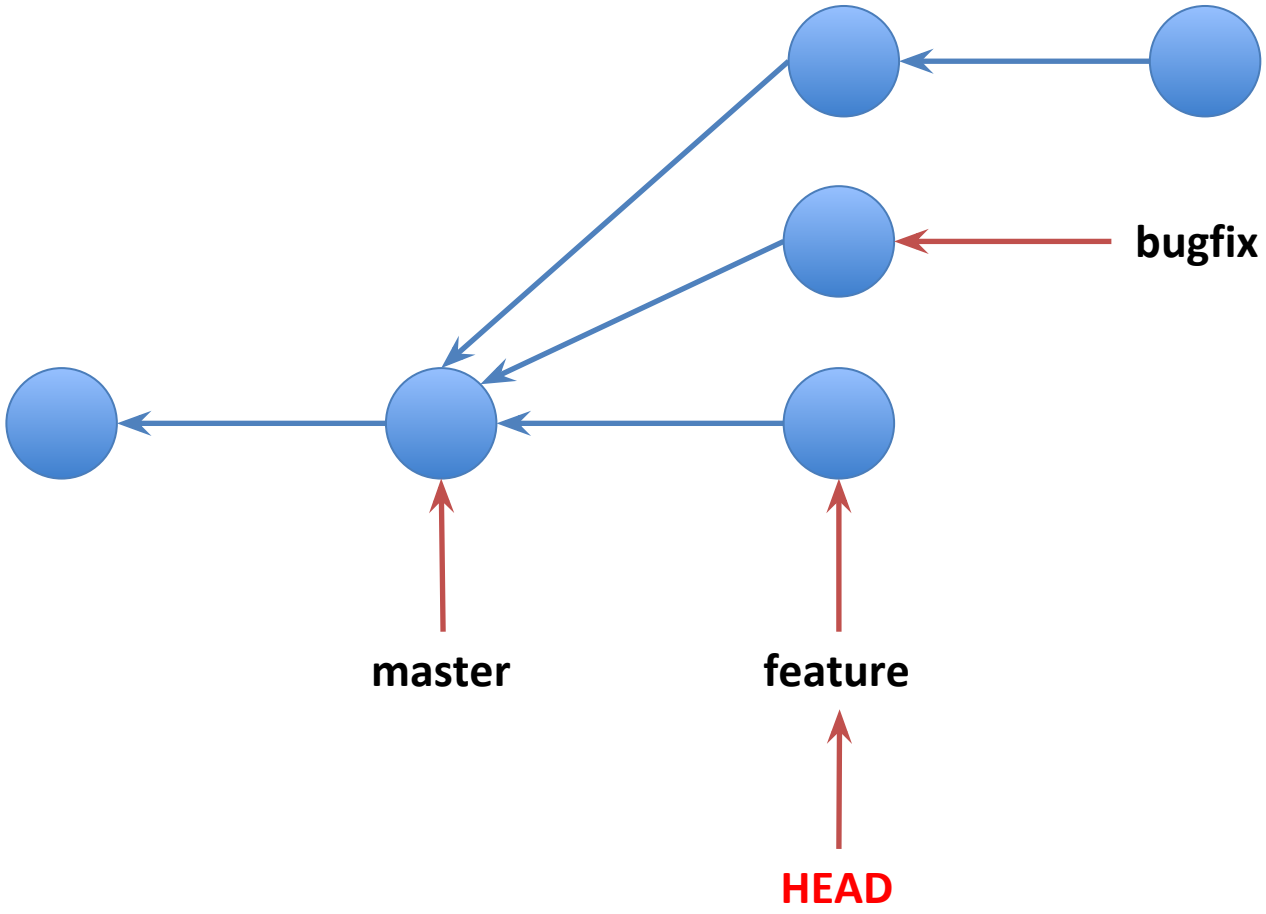
git checkout master



git commit



Quiz: Δημιούργησε τον γράφο



```
git init
git commit
git checkout -b bugfix
git commit
git checkout master
git checkout -b koko
git commit
git commit
git checkout master
git branch -D koko
git checkout -b feature
git commit
```

`git merge <branch>`

- Ενοποιεί το ιστορικό του branch που περνάς ως παράμετρο με το τωρινό branch
- Προσπαθεί να ενώσει τις αλλαγές στα αρχεία και από τα δύο branches
- Δημιουργεί ένα commit με 2 γονιούς:
  - Το τρέχον branch
  - Το branch που δίνεται ως παράμετρος

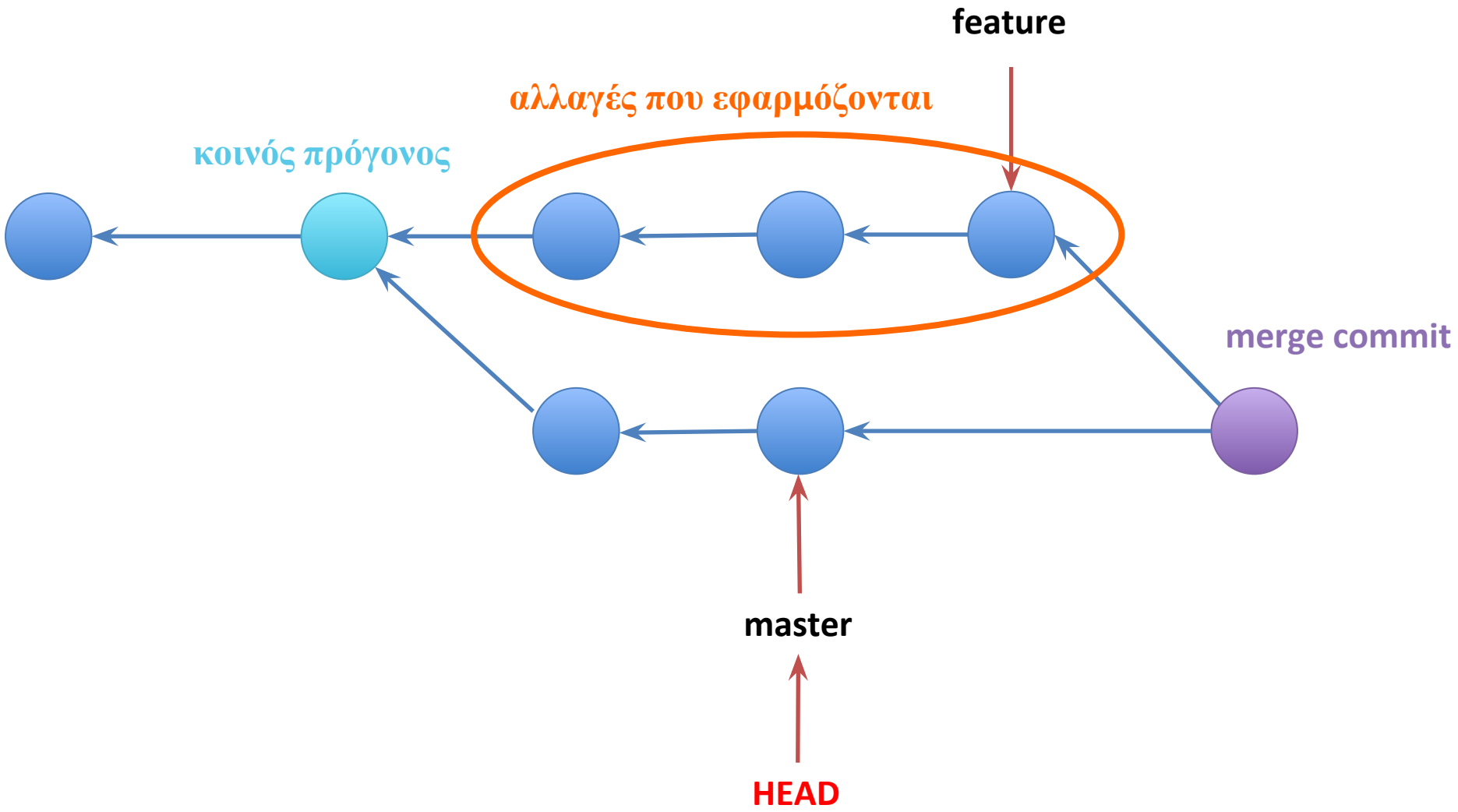
Αλγόριθμος merging

- Έστω ότι βρισκόμαστε στο master
- Τρέχουμε: `git merge feature`
- Αυτό δημιουργεί στο master τις αλλαγές που έγιναν εντωμεταξύ στο feature branch
- Δημιουργεί ένα νέο “merge commit” με δύο γονιούς:
  1. master
  2. feature

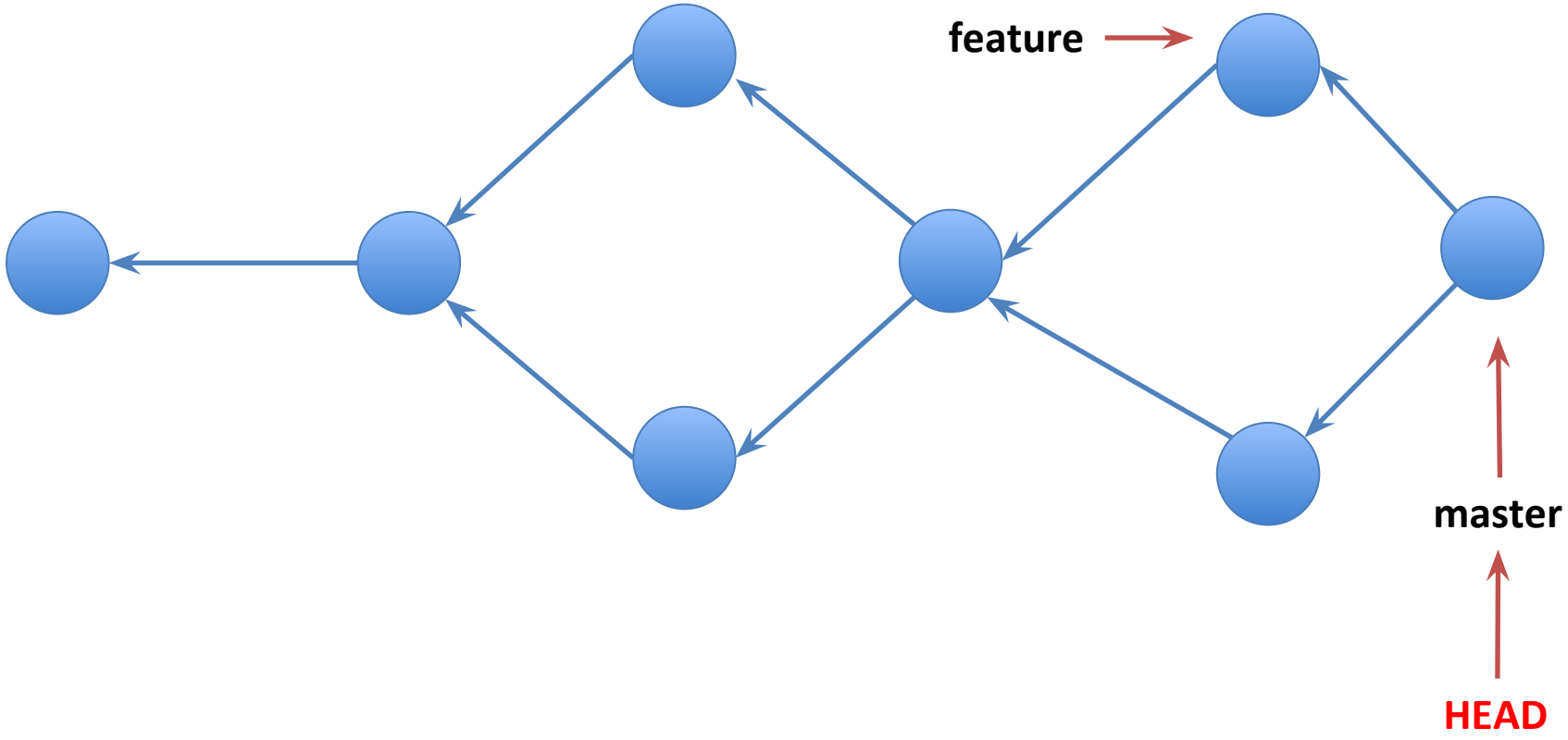
Αλγόριθμος merging

- Εντοπίζεται ο πιο πρόσφατος κοινός πρόγονος ανάμεσα σε master και feature
- Τα commits που πρέπει να εφαρμοστούν για να γίνει το merge είναι όλα τα commits ανάμεσα σε αυτόν τον πρόγονο και το feature branch
- Οι αλλαγές σε αυτά τα commits εφαρμόζονται στο master
- Το master μεταφέρεται στο νέο merge commit

(master) git merge feature



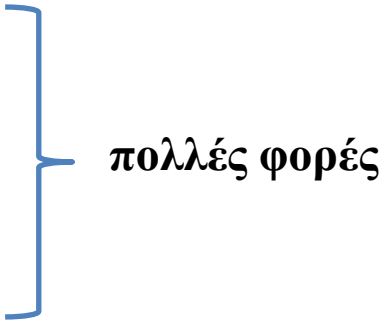
Quiz: Πώς γίνεται αυτό;



```
git commit
git commit
git checkout -b feature
git commit
git checkout master
git commit
git merge feature
git branch -d feature
git checkout -b feature
git commit
git checkout master
git commit
git merge feature
```

Branching workflow

git checkout master
git checkout -b feature
vim
git add
git commit
git checkout master
git merge feature
git branch -d feature



Stash

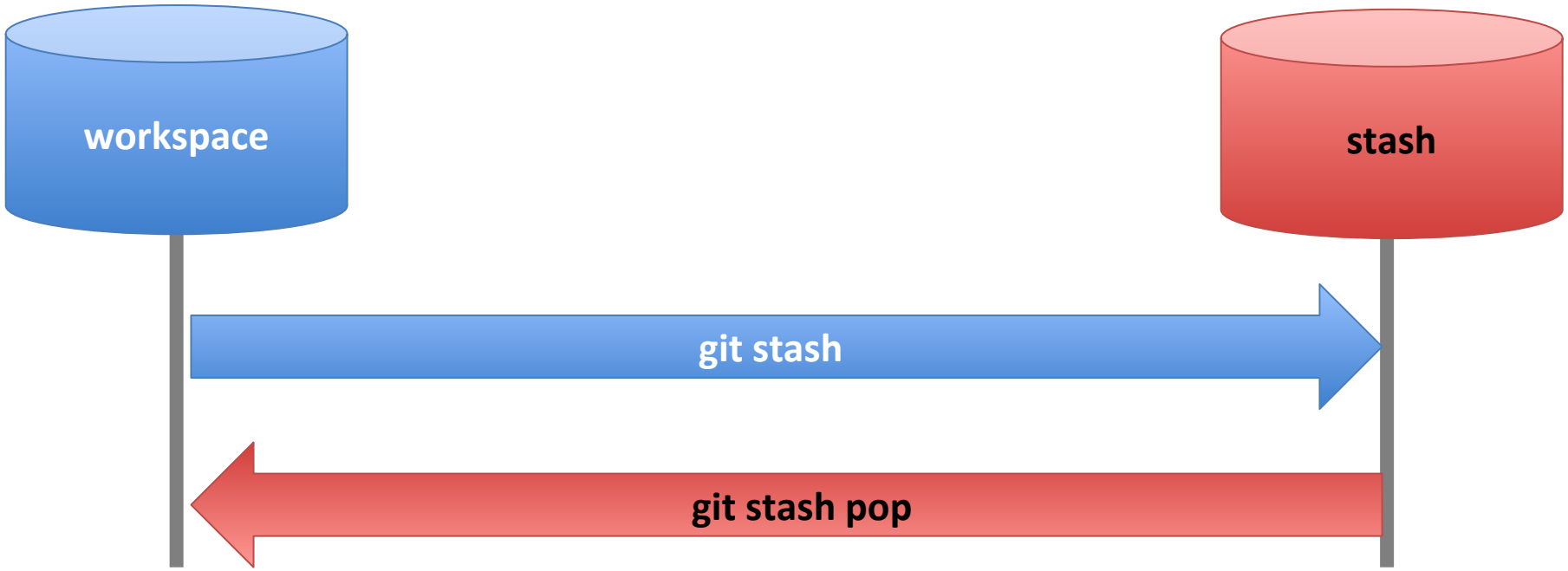
- Μερικές φορές χρειάζεται να αλλάξουμε branch για κάτι έκτακτο (π.χ. hotfix)
- Εντωμεταξύ μπορεί να έχουμε κάνει αλλαγές στο working copy μας
- Επιλογές:
  - commit… όμως δε θέλουμε να κάνουμε commit κάτι τσαπατσούλικο
  - checkout… όμως δε θέλουμε να χάσουμε τις αλλαγές μας

git stash

- Κρατάει στην άκρη τις αλλαγές:
  - στο working copy
  - στο staging area
- Καθαρίζει το working copy και το staging
- Πλέον μπορούμε να κάνουμε checkout ένα άλλο branch χωρίς να πρέπει να κάνουμε commit ή να χάσουμε τις αλλαγές μας

`git stash pop`

- Επανεφαρμόζει στο working copy τις αλλαγές που είναι αποθηκευμένες στο stash


workspace
stash
git stash
git stash pop

Remotes

- Ο κώδικας είναι συνεργατικό πράγμα
- Μία ομάδα χρειάζεται πολλά αντίγραφα του κώδικά της, ένα για κάθε προγραμματιστή
- Μερικές φορές χρειάζεται να ανταλλάξουμε κώδικα με αυτά τα αντίγραφα
- Κάθε αντίγραφο με το οποίο ανταλλάσσουμε κώδικα ονομάζεται “remote”

GitLab

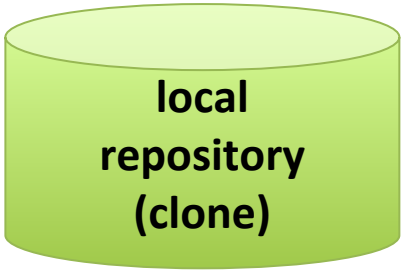
- Ένα website που μπορούμε να ανεβάσουμε αντίγραφα των repo μας και να συνεργαστούμε
- Προσφέρει εργαλεία για συνεργασία

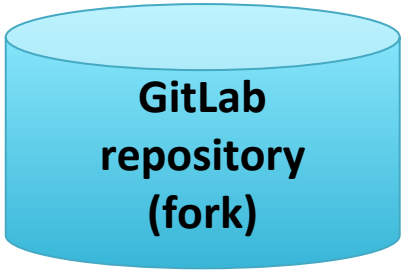
Ας συνεργαστούμε!

- Μπείτε στο http://anapgit.scanlab.gr/
- Πατήστε “create progect”
- Και συνεργαστείτε με άλλους

Remotes

- Αυτά τα δύο repos συνδέονται:
  - το clone στον υπολογιστή σου
  - το fork σου εντός του GitLab
- Το repo στον υπολογιστή σου έχει ως **remote** το repo σου στο GitLab





Remotes

- Κάθε αντίγραφο του repo έχει τα δικά του commits, branches, και ιστορικό
- Κάποια από αυτά μπορεί να είναι κοινά
- Κάθε remote έχει:
  - το δικό του URL
  - ένα όνομα

git remote

`git remote`

- Δείχνει ποια είναι τα remotes του repo μας

`git remote add <name> <url>`

- Προσθέτει ένα καινούργιο remote με το οποίο σκοπεύουμε να συνεργαστούμε

`git remote rm <name>`

- Διαγράφει το remote

origin

- Όταν κάνουμε git clone δημιουργείται αυτόματα ένα remote με το όνομα “origin” το οποίο είναι το δικό μας αντίγραφο του repo στο GitLab
- Αυτό είναι το remote που θα χρησιμοποιείς περισσότερο

git push

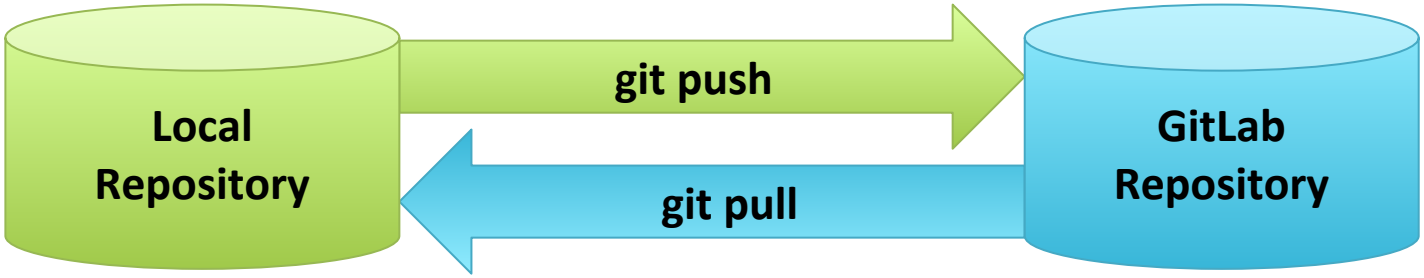
`git push <remote> <branch>`

- Στέλνει τα commits που έχουμε από το τοπικό ενεργό branch στο branch του remote που επιλέξαμε
- Έτσι δημοσιεύουμε τον κώδικά μας
- πχ: `git push origin master`

git pull

`git pull <remote> <branch>`

- Φέρνει στο τοπικό ενεργό branch τα commits που υπάρχουν στο branch του remote που επιλέξαμε
- Έτσι κατεβάζουμε τις αλλαγές των άλλων
- πχ: `git pull origin master`


Local
Repository
git push
git pull
GitLab
Repository

Remote workflow

git checkout master

git pull origin master

git checkout -b feature

vim && git add && git commit

git checkout master

git merge feature

git push origin master

Outdated

- Κάποιος άλλος έκανε push commits εντωμεταξύ
- Για να μην γράψουμε **πάνω από τις αλλαγές των άλλων**, το git ζητάει να κάνουμε pull
- Με το pull γίνεται το εξής:
  - Κατεβαίνει ο νέος κώδικας (git fetch)
  - Ενοποιείται με τις δικές μας αλλαγές (git merge)
- Στη συνέχεια μπορούμε να κάνουμε push

git

Ευχαριστούμε!!