Git & GitHub

Δημήτριος Παρασχάς
Κωνσταντίνος Κανελλής

IEEE Student Branch
University of Thessaly

First things first...

- Δανειστήκαμε υλικό από git-class.gr και το τροποποιήσαμε κατάλληλα για την σημερινή παρουσίαση
  - Σας ευχαριστούμε!
- Μπορείτε να βρείτε υλικό για ακόμα πιο εξειδικευμένα θέματα στο Git (που δεν θα καλύψουμε σήμερα)
  - Σας προτείνουμε να τα δείτε!

Τι θα μάθουμε

- Τι είναι το git
- Βασική χρήση git
- Πώς δουλεύουμε τοπικά με git
- Πώς δουλεύουμε απομακρυσμένα με git
- Συνεργασία μέσω git & GitHub
- git & GitHub workflow

Το πρόβλημα

- Ως προγραμματιστές έχουμε κάποιες ανάγκες
  - Νέος κώδικας μερικές φορές είναι buggy
  - Δουλεύουμε πολλοί ταυτόχρονα στον ίδιο κώδικα
  - Διαγράφουμε κώδικα που μπορεί να χρειαστεί ξανά
  - Χρειαζόμαστε backups για τη δουλειά μας

Πώς λύνουμε αυτά τα προβλήματα;

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

EVERY DESIGNER IN THIS WORLD

EVERY DESIGNER IN THIS WORLD

new.**psd**

newfinal.**psd**

EVERY DESIGNER IN THIS WORLD

new.**psd**

newfinal.**psd**

newfinalfinal.**psd**

EVERY DESIGNER IN THIS WORLD

new.**psd**

newfinal.**psd**

newfinalfinal.**psd**

newfinalestfinal.**psd**

EVERY DESIGNER IN THIS WORLD

new.**psd**

newfinal.**psd**

newfinalfinal.**psd**

newfinalestfinal.**psd**

newfinalestfinal forsure.**psd**

EVERY DESIGNER IN THIS WORLD

new.**psd**

newfinal.**psd**

newfinalfinal.**psd**

newfinalestfinal.**psd**

newfinalestfinal forsure.**psd**

newfinalestfuckthis shitfinal.**psd**

Version Control

- Μπορούμε να ...
  - κρατάμε πολλές εκδόσεις των αρχείων μας
  - αναιρούμε αλλαγές
  - συνεργαζόμαστε με άλλους
  - κρατάμε αντίγραφα ασφαλείας
  - μοιραζόμαστε εύκολα κώδικα με μια ομάδα
  - ξέρουμε ποια είναι η “τελευταία” έκδοση

Ιστορία του version control

- Centralized version control systems (CSV, SVN)
- git - 2005

  Distributed version control system
- GitHub - 2008

  Συνεργατικό περιβάλλον version control

Τι είναι το Git?

- Πρόγραμμα που τρέχεις στον υπολογιστή σου
- Εργαλείο στο terminal
- Το οποίο διαχειρίζεται τον κώδικά σου

Εγκατάσταση του git

- Linux (Debian, Ubuntu)

  apt-get install git

- Mac

  Τρέξε git και ακολούθησε τις οδηγίες εγκατάστασης

- Windows

  format C:

Εγκατάσταση του git

- Linux (Debian, Ubuntu)

  apt-get install git
- Mac

  Τρέξε git και ακολούθησε τις οδηγίες εγκατάστασης
- Windows
  - Κατέβασμα από το https://git-scm.com/download
  - Εγκατάσταση
  - Τρέξε το git CMD

Ας ξεκινήσουμε ένα project!

- Ανοίξτε όλοι τα laptops σας

Ρύθμιση του git

- Πρέπει να πεις στο git ποιος είσαι
  - `git config --global [-e]`
  - Αλλάζει τις ρυθμίσεις του git
  - Το όνομα και το email σου συνοδεύουν τον δημόσιο κώδικά σου

```
dp@alpha ~ $ git config --global user.name "Dimitrios Paraschas"
dp@alpha ~ $ git config --global user.email "paraschas@gmail.com"
```

add license
paraschas committed on Mar 9, 2014
775dfa0

add the slides of the presentation
paraschas committed on Mar 9, 2014
e6efa4d

remove capitalization
paraschas committed on Mar 9, 2014
cbc6e36

Commits on Mar 8, 2014

Remove slide example
kkanellis committed on Mar 8, 2014
fb865ad

Added list comprehension workshop
kkanellis committed on Mar 8, 2014
08b0421

Added google exercises
kkanellis committed on Mar 8, 2014
9d50aec

Fixed paths
kkanellis committed on Mar 8, 2014
f47e02e

`git init`

- Άνοιξε τη γραμμή εντολών
- Δημιούργησε ένα φάκελο
- Μέσα στο φάκελο τρέξε:
  `git init`
- Αρχικοποιεί ένα git repository για τη διαχείριση του κώδικά σου

```
dp@alpha git-example $ git init
Initialized empty Git repository in /home/dp/git-GitHub/git-example/.git/
dp@alpha git-example (master #) $
```

Ο φάκελος .git

- Υπάρχει μέσα στο project μας
- Μέσα αποθηκεύει δεδομένα και μετα-δεδομένα σχετικά με το repository μας
- Ένα project που διαχειρίζεται από το git λέγεται “repository” ή “repo”

```
dp@alpha git-example (master #) $ git status
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)
dp@alpha git-example (master #) $
```

`git status`

- Δείχνει την κατάσταση του project
- Τρέξτο αν δεν ξέρεις σε τι κατάσταση είναι το repository σου

Ας δημιουργήσουμε ένα αρχείο

- Δημιούργησε ένα απλό αρχείο κειμένου

```
dp@alpha git-example (master #) $ vim README.md
dp@alpha git-example (master #) $ cat README.md
Hello, git world!
dp@alpha git-example (master #) $ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        README.md

nothing added to commit but untracked files present (use "git add" to track)
dp@alpha git-example (master #) $ git add README.md
dp@alpha git-example (master #) $ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   README.md

dp@alpha git-example (master #) $
```

`git add <file>`

- Ζητάει από το git να παρακολουθεί ένα νέο αρχείο
  - Πρέπει να ακολουθείται από “commit”
  - Παίρνει ως παράμετρο το νέο αρχείο
  - ή ένα γενικό μοτίβο
  - . όλα τα αρχεία
  - '*.c' όλα τα αρχεία .c

git commit

- Δημιουργεί ένα “commit”

```
dp@alpha git-example (master #) $ git commit -m "add README"
[master (root-commit) 51d0013] add README
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
dp@alpha git-example (master) $
```

Τι είναι ένα “commit αντικείμενο”;

- Ένα στιγμιότυπο του project μας
  - Το σύνολο των αρχείων και φακέλων του project, με τα περιεχόμενά τους σε μία χρονική στιγμή
- Περιλαμβάνει ένα περιγραφικό μήνυμα
- Καταγράφει μετα-δεδομένα
  - Ημερομηνία και ώρα
  - Δημιουργό
- Έχει ένα μοναδικό αναγνωριστικό
  - π.χ. 51d0013f85982c968f7cbbb79ed992af130fed59

Κατάσταση μετά το commit

```
dp@alpha git-example (master #) $ git commit -m "add README"
[master (root-commit) 51d0013] add README
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
dp@alpha git-example (master) $ git status
On branch master
nothing to commit, working directory clean
dp@alpha git-example (master) $
```

Ας κάνουμε μια αλλαγή

- Άλλαξε μια λέξη στο αρχείο

```
dp@alpha example-repo (master) $ vim README.md
dp@alpha example-repo (master *) $ git status
On branch 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")
dp@alpha example-repo (master *) $ git diff
```

```
 -- a/README.md
 ++ b/README.md
```

```
@@ -1 +1 @@
 Hello, git world!
 Hello, GitHub world!
dp@alpha example-repo (master *) $ git commit -a -m "replace git with GitHub"
[master 78c429e] replace git with GitHub
 1 file changed, 1 insertion(+), 1 deletion(-)
dp@alpha example-repo (master) $
```

`git diff`

- Δείχνει τι άλλαξε στο project 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 ως παράμετρο συγκεκριμένα αρχεία που μας ενδιαφέρουν

`git commit -a`

- Κάνει commit όλες τις αλλαγές που έχουν γίνει στα αρχεία του repository που γνωρίζει και διαχειρίζεται το git

Ρύθμιση `core.editor`

- Λέμε στο git ποιον text editor θέλουμε να χρησιμοποιούμε

```
dp@alpha git-example (master) $ git config --global core.editor "vim"
```

ή

```
dp@alpha git-example (master) $ git config --global core.editor "kate"
```

Ας κάνουμε ακόμα μία αλλαγή

- Κάνε μια άλλη αλλαγή στο αρχείο

```
dp@alpha git-example (master) $ vim README.md
dp@alpha git-example (master *) $ git diff
```

```
 -- a/README.md
 ++ b/README.md
```

```
@@ -1 +1 @@
 Hello, GitHub world!
 Hello, git and GitHub world!
dp@alpha git-example (master *) $ git commit -a
[master eb96848] add git back
 1 file changed, 1 insertion(+), 1 deletion(-)
dp@alpha git-example (master) $
```

Βασικό git workflow

- `vim README.md`
- `[git status]`
- `git diff`
- `git commit -a`

Ας γράψουμε ένα πρόγραμμα

- Γράψε ένα απλό πρόγραμμα σε C

Ας γράψουμε ένα πρόγραμμα

- Γράψε ένα απλό πρόγραμμα σε C

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  printf("Hello World!\n");

  return 0;
}
```

```
dp@alpha git-example (master) $ vim advanced-app.c
dp@alpha git-example (master) $ gcc -Wall -g advanced-app.c -o advanced-app
dp@alpha git-example (master) $ ./advanced-app
Hello World!
dp@alpha git-example (master) $ git add advanced-app.c
dp@alpha git-example (master +) $ git commit -a
[master a0f3bd1] add an advanced C application
 1 file changed, 7 insertions(+)
 create mode 100644 advanced-app.c
dp@alpha git-example (master) $ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        advanced-app

nothing added to commit but untracked files present (use "git add" to track)
dp@alpha git-example (master) $
```

```
dp@alpha git-example (master) $ vim .gitignore
dp@alpha git-example (master) $ cat .gitignore
advanced-app
dp@alpha git-example (master) $ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
dp@alpha git-example (master) $ git add .gitignore
dp@alpha git-example (master +) $ git commit -a
[master 9bb5809] add .gitignore
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore
dp@alpha git-example (master) $ git status
On branch master
nothing to commit, working directory clean
dp@alpha git-example (master) $ ls -Ahl
total 28K
-rwxrwxr-x 1 dp dp 9.5K Mar 24 19:21 advanced-app
-rw-rw-r-- 1 dp dp  102 Mar 24 19:21 advanced-app.c
drwxrwxr-x 8 dp dp 4.0K Mar 24 19:23 .git
-rw-rw-r-- 1 dp dp   13 Mar 24 19:22 .gitignore
-rw-rw-r-- 1 dp dp   29 Mar 24 19:16 README.md
dp@alpha git-example (master) $
```

.gitignore

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

Παράδειγμα .gitignore

```
*.swp
*.pyc
*.class
*.o
*.out
*.log
```

Ας γράψουμε το πρόγραμμα σε Python

```
print("Hello World!")
```

```
dp@alpha git-example (master) $ vim elite-app.py
dp@alpha git-example (master) $ cat elite-app.py
print("Hello World!")
dp@alpha git-example (master) $ python elite-app.py
Hello World!
dp@alpha git-example (master) $ git add elite-app.py
dp@alpha git-example (master +) $ git rm advanced-app.c
rm 'advanced-app.c'
dp@alpha git-example (master +) $ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    advanced-app.c
        new file:   elite-app.py

dp@alpha git-example (master +) $ git commit -a
[master 10b4a5c] Python FTW!
 2 files changed, 1 insertion(+), 7 deletions(-)
 delete mode 100644 advanced-app.c
 create mode 100644 elite-app.py
dp@alpha git-example (master) $ git mv elite-app.py leet-app.py
dp@alpha git-example (master +) $ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        renamed:    elite-app.py -> leet-app.py

dp@alpha git-example (master +) $ git commit -a
[master 71f857f] fix spelling
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename elite-app.py => leet-app.py (100%)
dp@alpha git-example (master) $
```

`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
- ισοδύναμο με:
  - `mv <old path> <new path>`
  - `git add <new path>`
  - `git rm <old path>`

`git add <file>`

- Ζητάει από το git να παρακολουθεί ένα νέο αρχείο
- Προετοιμάζει τις αλλαγές ενός αρχείου να γίνουν commit
- Πρέπει να ακολουθείται από “commit”

`git diff --staged`

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

`git help <command>`

- Βοήθεια για κάποιο git command
- για παράδειγμα:
  - `git help add`
  - `git help commit`

Ιστορικό

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

`git log`

- Δείχνει το ιστορικό
- Με το log βλέπουμε:
  - Τη λίστα του ιστορικού με όλα τα commits
  - Ποιος έκανε το κάθε commit
  - Περιγραφή
  - Αντίστροφη χρονολογική σειρά

git log

```
* commit a525c0b7fc40dccee45e2cad342bd39f86577697
| Author: Dimitrios Paraschas <paraschas@gmail.com>
| Date:   2016-03-24 16:33:27 +0200
|
|     replace git with GitHub
|
* commit 51d0013f85982c968f7cbbb79ed992af130fed59
  Author: Dimitrios Paraschas <paraschas@gmail.com>
  Date:   2016-03-24 12:36:33 +0200

      add README
```

git log

Αναγνωριστικό του commit

```
* commit a525c0b7fc40dccee45e2cad342bd39f86577697
| Author: Dimitrios Paraschas <paraschas@gmail.com>
| Date:   2016-03-24 16:33:27 +0200
|
|     replace git with GitHub
|
* commit 51d0013f85982c968f7cbbb79ed992af130fed59
  Author: Dimitrios Paraschas <paraschas@gmail.com>
  Date:   2016-03-24 12:36:33 +0200

      add README
```

git log

```
* commit a525c0b7fc40dccee45e2cad342bd39f86577697
| Author: Dimitrios Paraschas <paraschas@gmail.com>
| Date:   2016-03-24 16:33:27 +0200
|
|     replace git with GitHub
|
* commit 51d0013f85982c968f7cbbb79ed992af130fed59
  Author: Dimitrios Paraschas <paraschas@gmail.com>
  Date:   2016-03-24 12:36:33 +0200

      add README
```

Πρώτο commit

git log

Δεύτερο commit

```
* commit a525c0b7fc40dccee45e2cad342bd39f86577697
| Author: Dimitrios Paraschas <paraschas@gmail.com>
| Date:   2016-03-24 16:33:27 +0200
|
|     replace git with GitHub
|
* commit 51d0013f85982c968f7cbbb79ed992af130fed59
  Author: Dimitrios Paraschas <paraschas@gmail.com>
  Date:   2016-03-24 12:36:33 +0200

      add README
```

Πρώτο commit

Ένα καλύτερο git log: `git lg`

Το git είναι παραμετροποιήσιμο. Μπορούμε να φτιάξουμε τις δικές μας εντολές.

π.χ. διαδεδομένη η εντολή git lg

```
git config --global alias.lg "log --
decorate --all --graph --abbrev-commit
--pretty=format:'%C(yellow bold)%h%
Creset%C(auto)%d%Creset %s %Cgreen(%cr)
%C(bold blue)<%an>%Creset'"
```

git lg

```
dp@alpha git-example (master) $ git lg
* 71f857f (HEAD, master) fix spelling (2 hours ago) <Dimitrios Paraschas>
* 10b4a5c Python FTW! (2 hours ago) <Dimitrios Paraschas>
* 9bb5809 add .gitignore (33 hours ago) <Dimitrios Paraschas>
* a0f3bd1 add an advanced C application (33 hours ago) <Dimitrios Paraschas>
* eb96848 add git back (34 hours ago) <Dimitrios Paraschas>
* a525c0b replace git with GitHub (2 days ago) <Dimitrios Paraschas>
* 51d0013 add README (2 days ago) <Dimitrios Paraschas>
dp@alpha git-example (master) $
```

`git show`

- Μπορεί να δείξει διάφορα αντικείμενα
- 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 (το diff)

git show

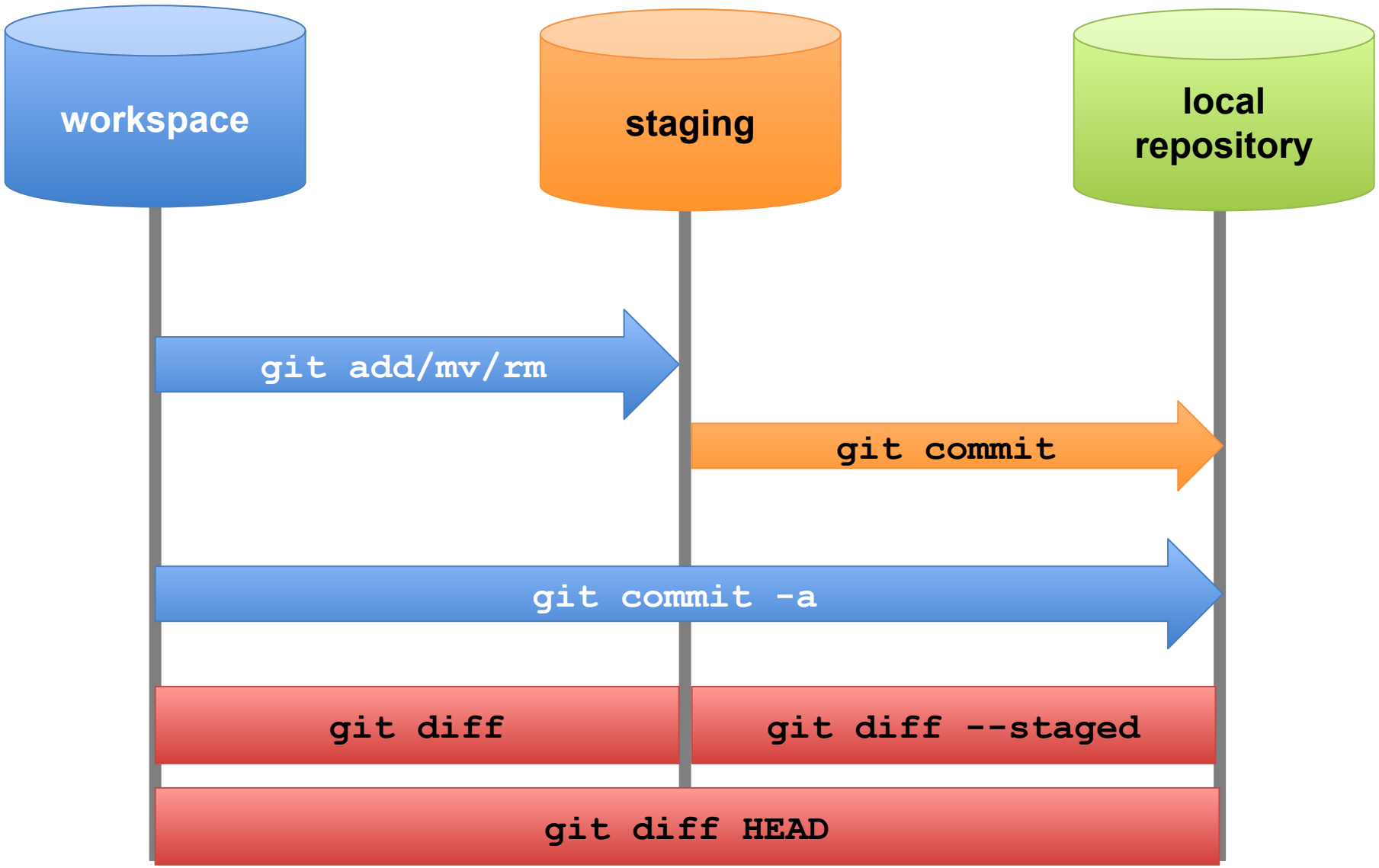
```
dp@alpha git-example (master) $ git show a525c
commit a525c0b7fc40dccee45e2cad342bd39f86577697
Author: Dimitrios Paraschas <paraschas@gmail.com>
Date:   Thu Mar 24 16:33:27 2016 +0200

    replace git with GitHub

---- a/README.md
++++ b/README.md

@ README.md:1 @
-Hello, git world!
+Hello, GitHub world!
dp@alpha git-example (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. a525c0b7fc40dccee45e2cad342bd39f86577697
- Μπορούμε να αναφερθούμε και με ένα πρόθεμα (τουλάχιστον 4 χαρακτήρες)
  - a525c
  - Αρκεί να είναι μοναδικό μέσα στο repo

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

Quiz: Τι δείχνει αυτό το `git status`?

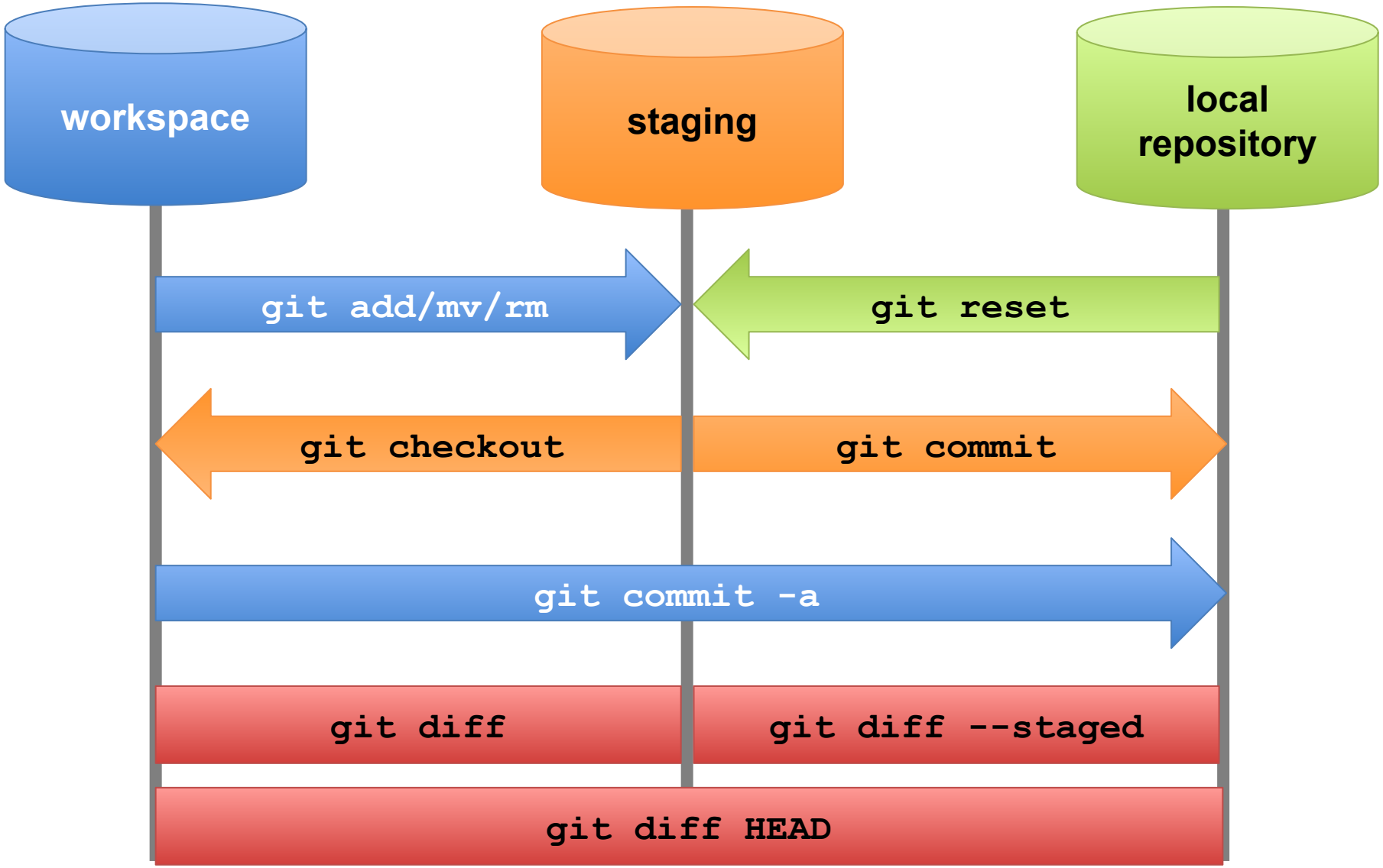
```
dp@alpha git-example (master *+) $ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   leet-app.py

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

dp@alpha git-example (master *+) $
```

`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

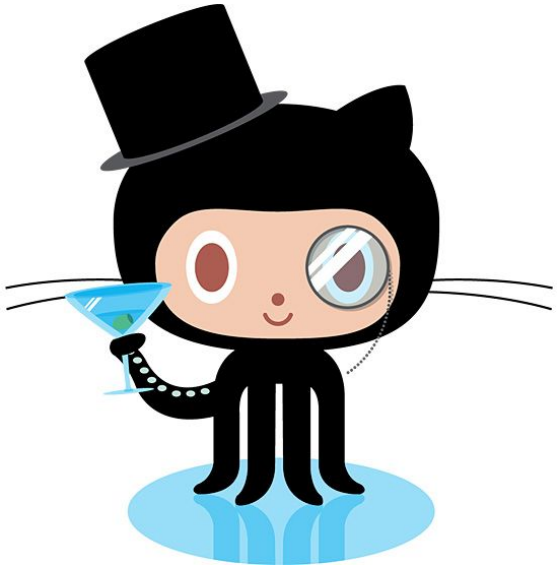

workspace
staging
local repository
git add/mv/rm
git reset
git checkout
git commit
git commit -a
git diff
git diff --staged
git diff HEAD

Μάθαμε

- Τοπική χρήση του git
  - init
  - add/rm/mv
  - status
  - diff
  - commit
  - log/lg/show
  - checkout

Διάλειμμα

- φτιάξτε GitHub accounts
- γνωριστείτε μεταξύ σας!



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

```
dp@alpha git-example (master) $ git branch
* master
dp@alpha git-example (master) $ git branch improve-readme
dp@alpha git-example (master) $ git branch
  improve-readme
* master
dp@alpha git-example (master) $ git checkout improve-readme
Switched to branch 'improve-readme'
dp@alpha git-example (improve-readme) $ git branch
* improve-readme
  master
dp@alpha git-example (improve-readme) $
```

`git checkout -b <branch>`

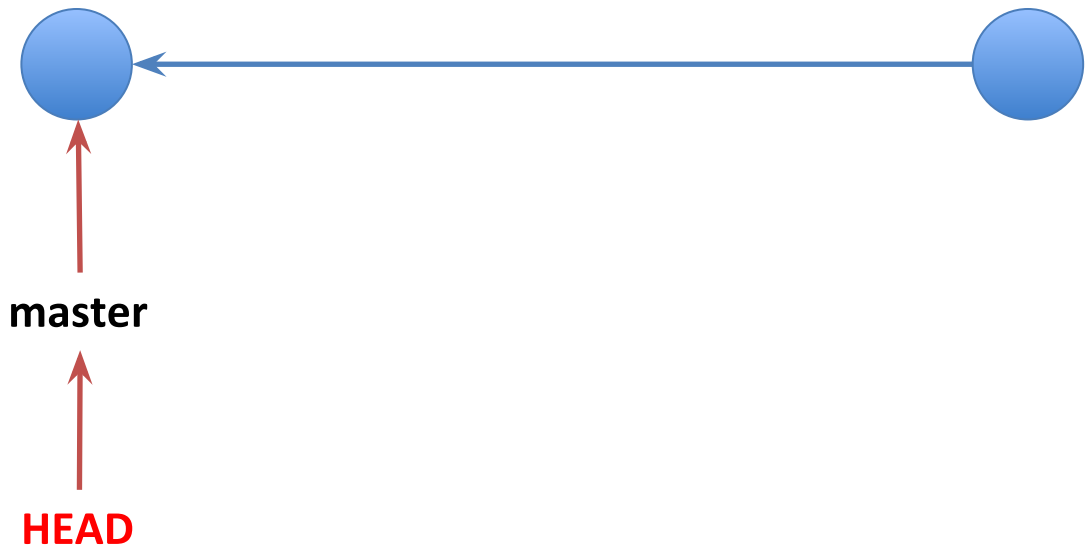
- Δημιουργεί νέο branch και αλλάζει το τρέχον branch σε <branch>
- `git checkout -b <branch>` σημαίνει:
  - `git branch <branch>`
  - `git checkout <branch>`

Ο γράφος του git

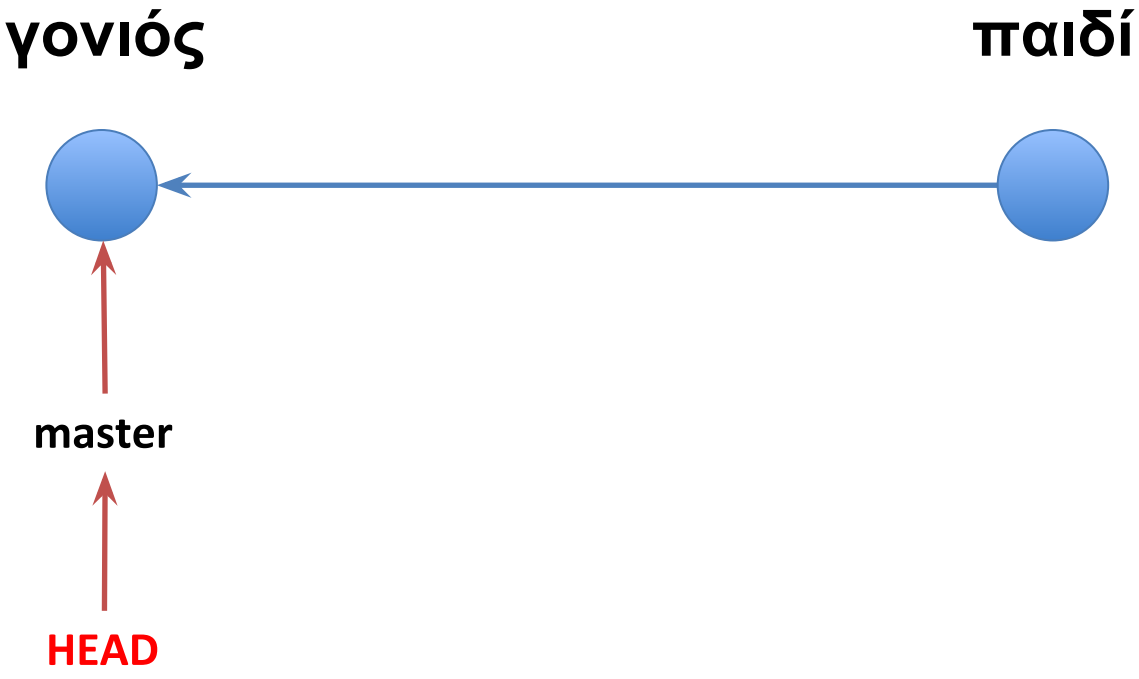
- Το git είναι ένα **σύστημα επεξεργασίας γράφων**
- **Κάθε commit είναι ένας κόμβος**
- Κάθε commit έχει **γονιό** το προηγούμενο commit του
- Ένα branch δείχνει σε ένα commit
- Το HEAD δείχνει στο τρέχον branch
- Ένα branch επιτρέπει τη δημιουργία **διακλαδώσεων** στο γράφο

git commit

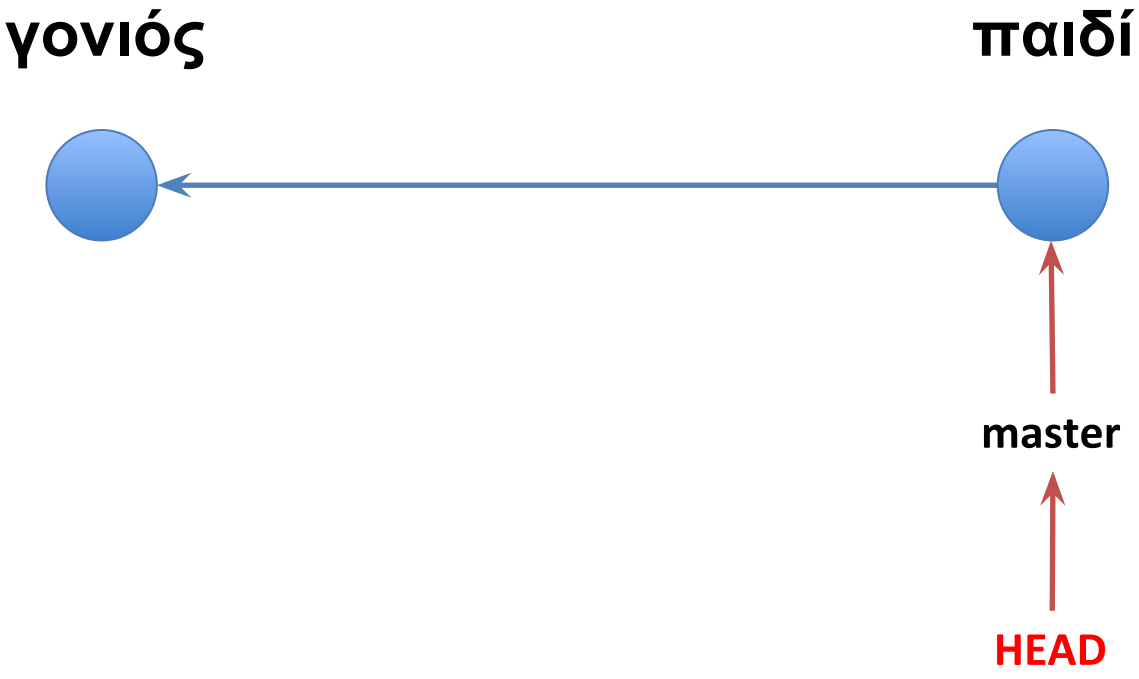
git commit



git commit

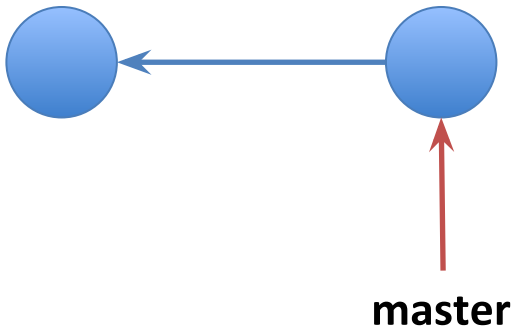


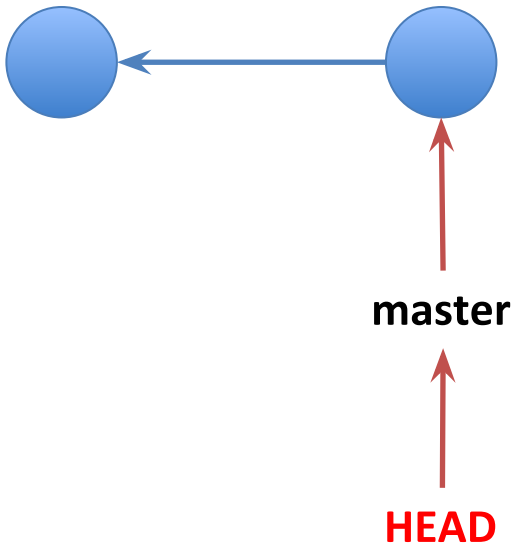
git commit



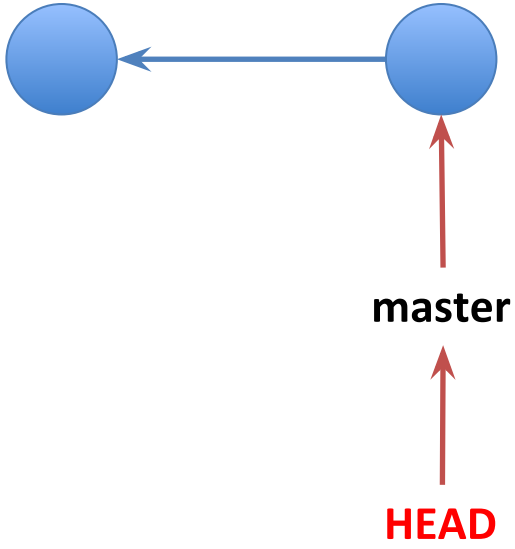
`git commit`

- 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

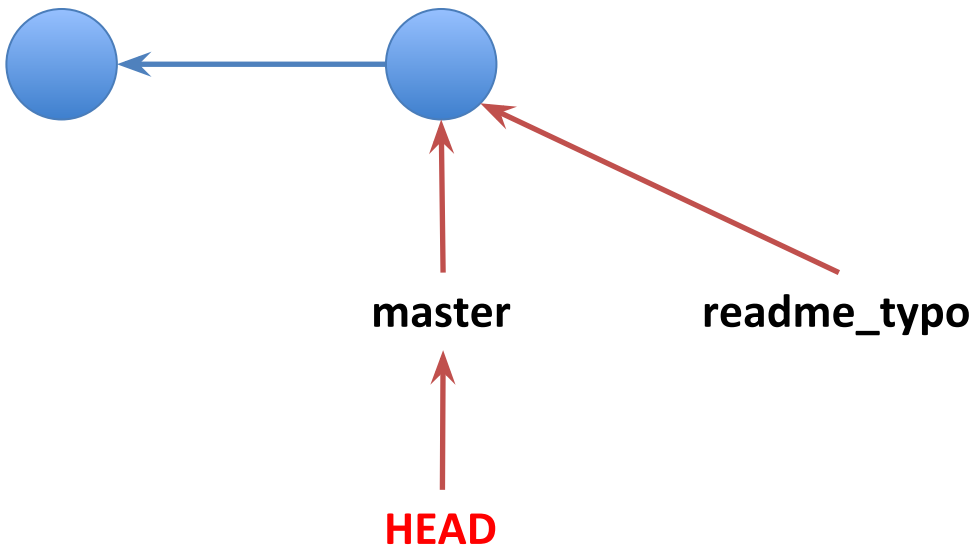

master

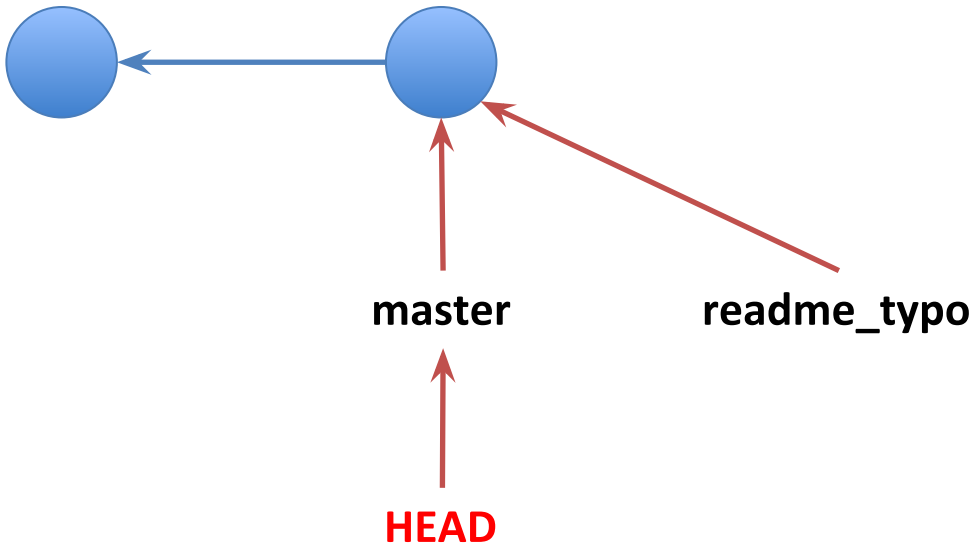

master
HEAD

`git branch readme_typo`

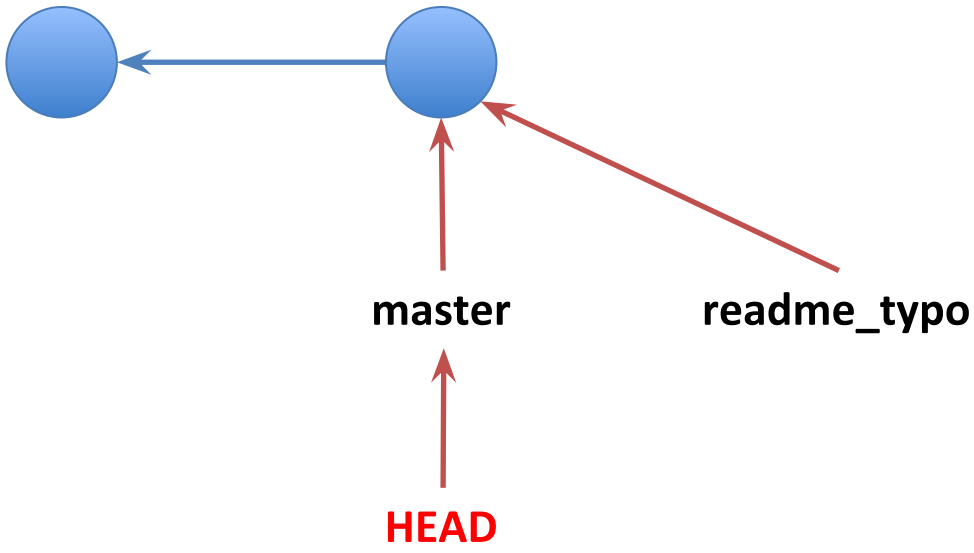


`git branch readme_typo`

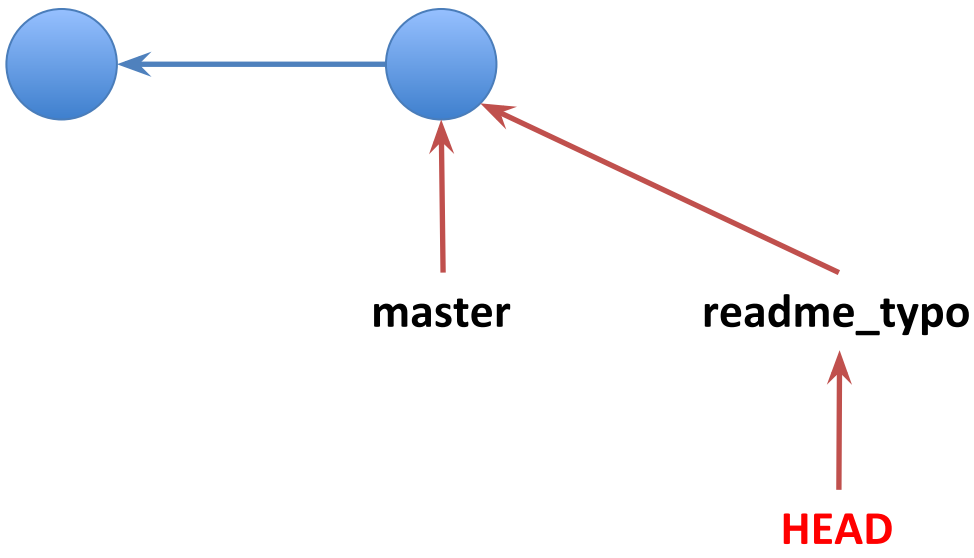


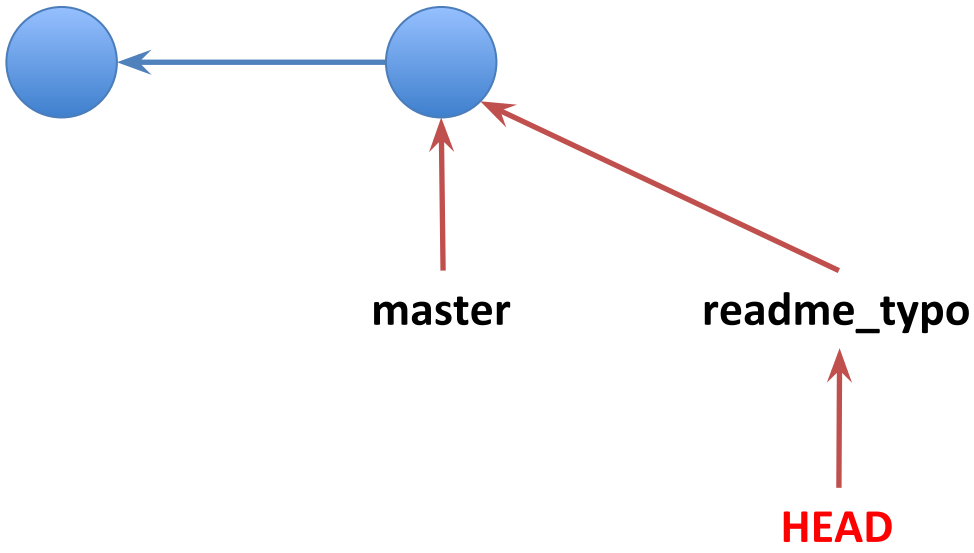

master
readme_typo
HEAD

`git checkout readme_typo`

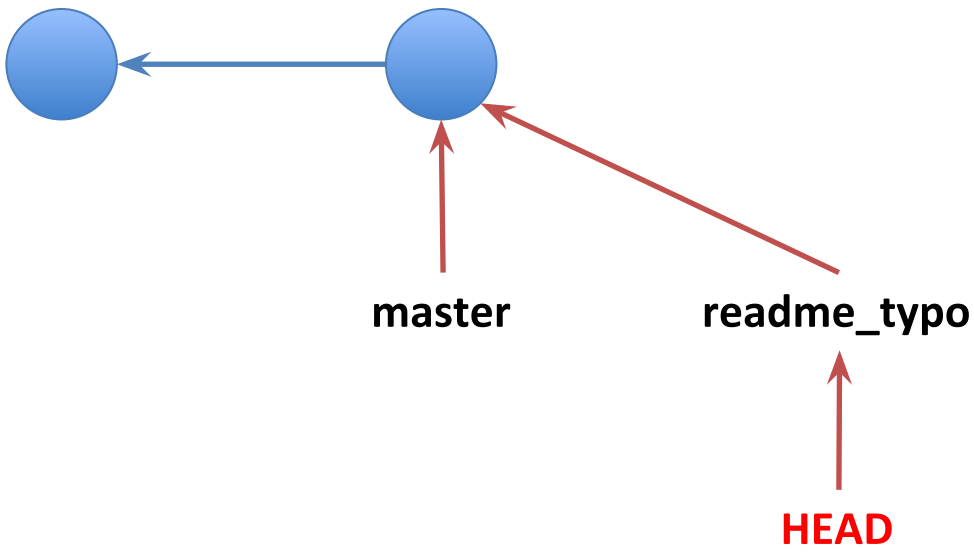


git checkout readme_typo

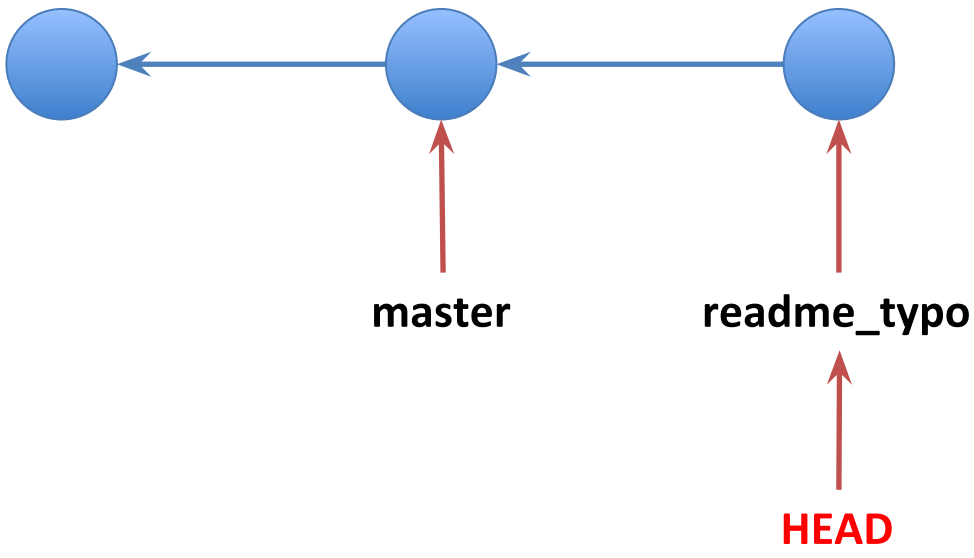


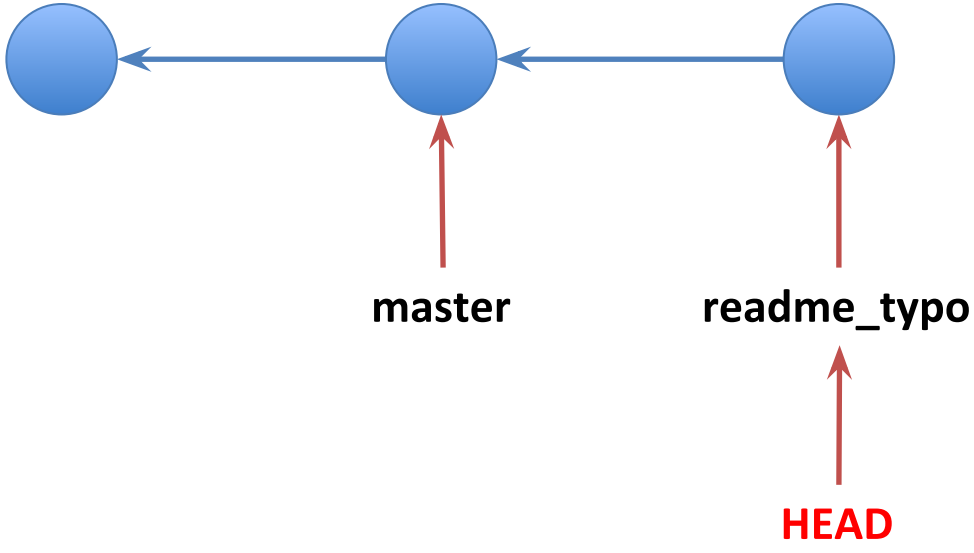

master
readme_typo
HEAD

git commit

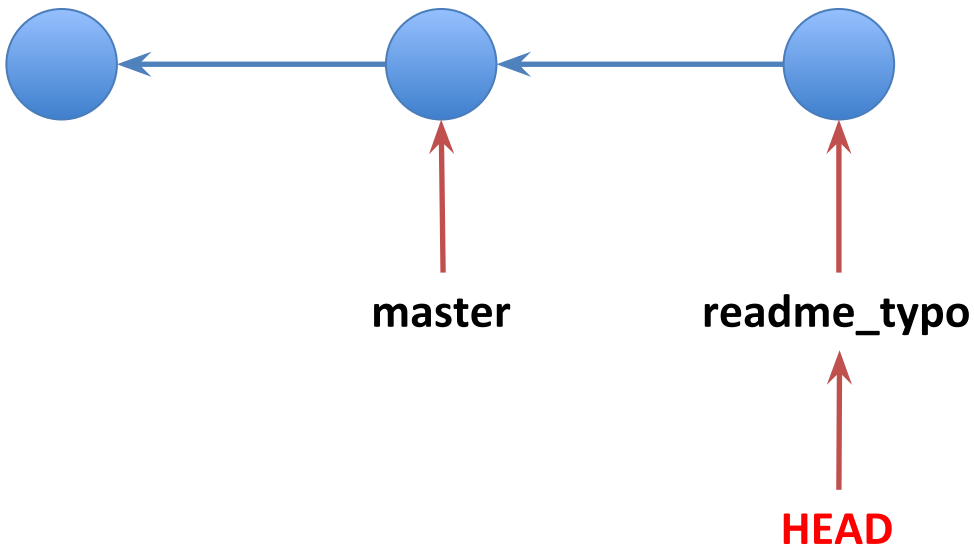


git commit

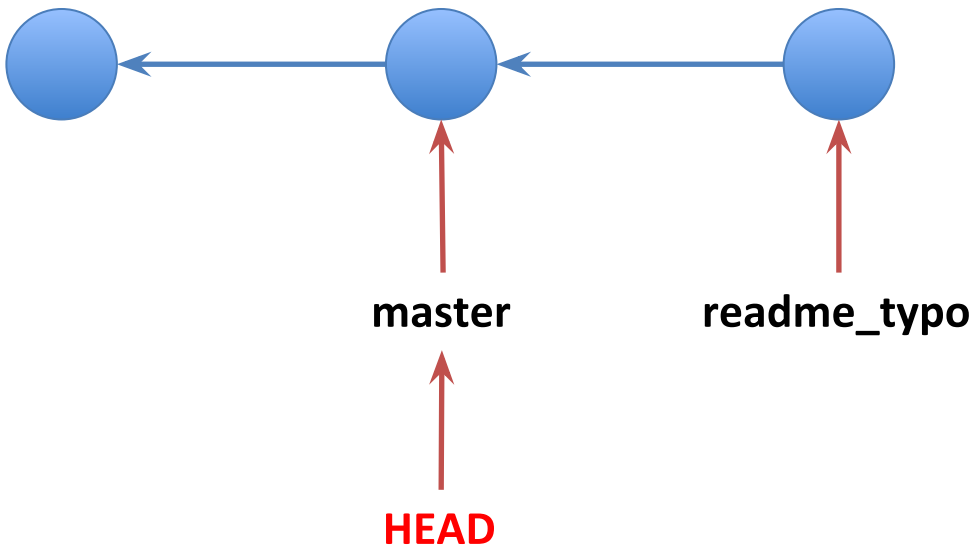


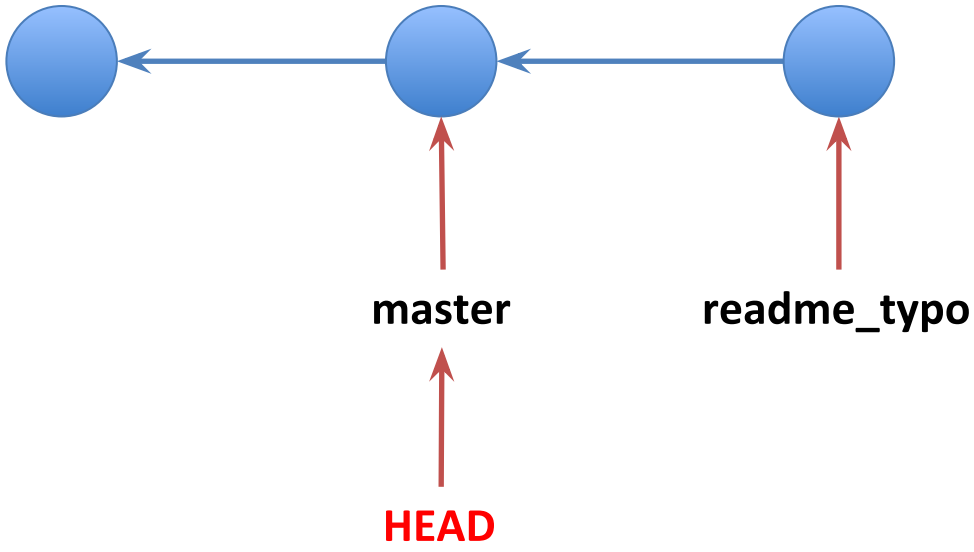

master
readme_typo
HEAD

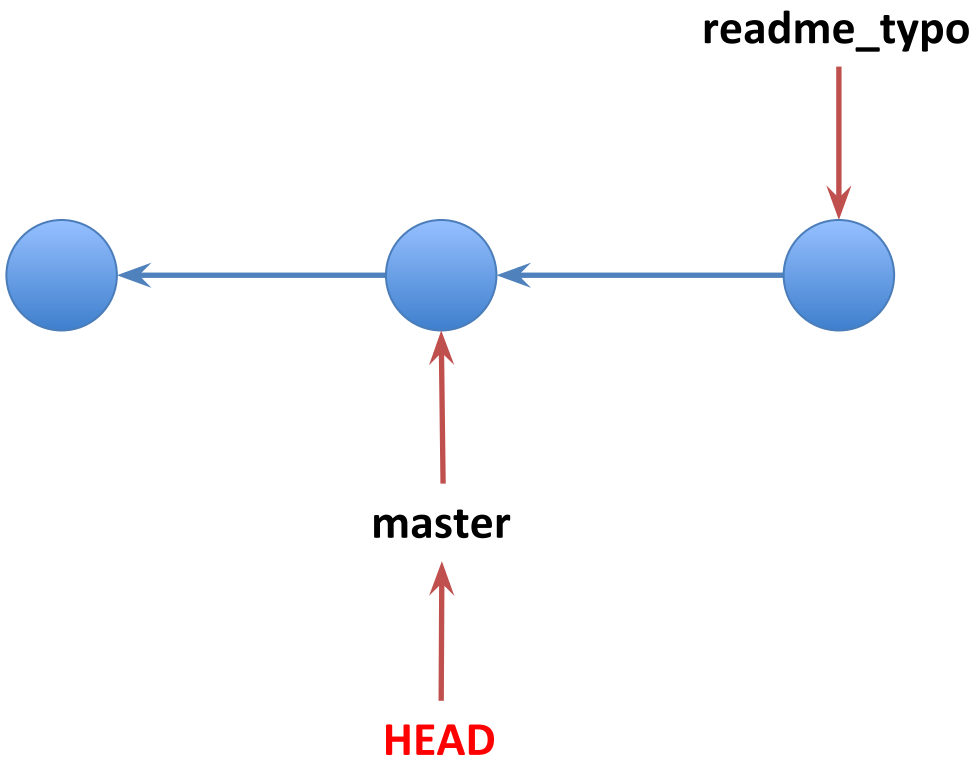
git checkout master



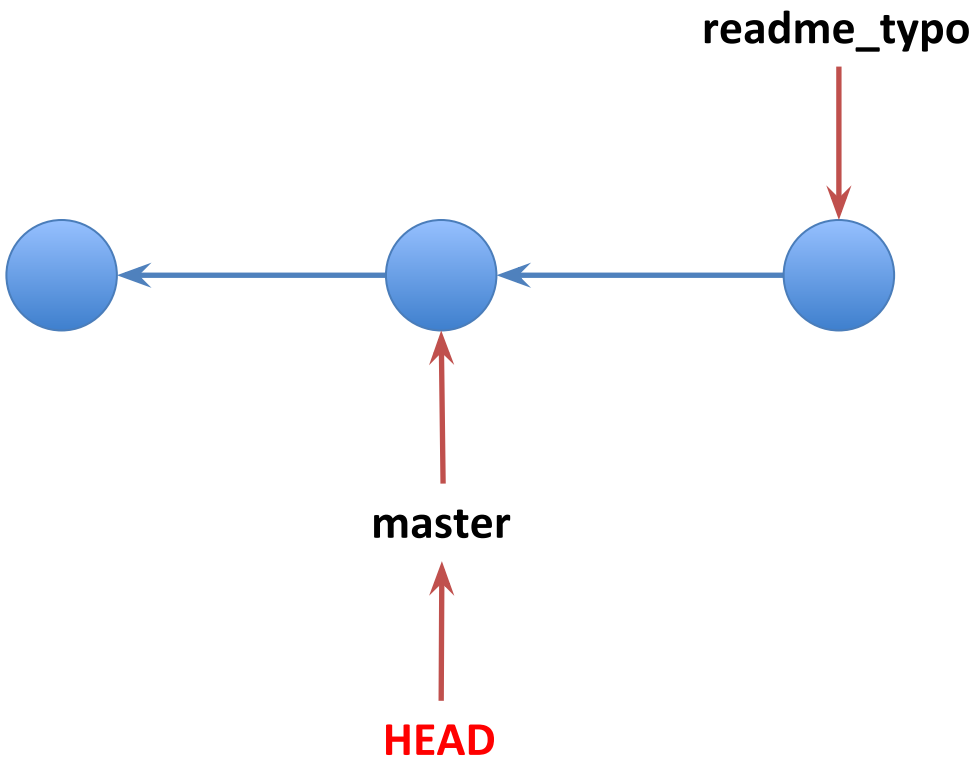
git checkout master



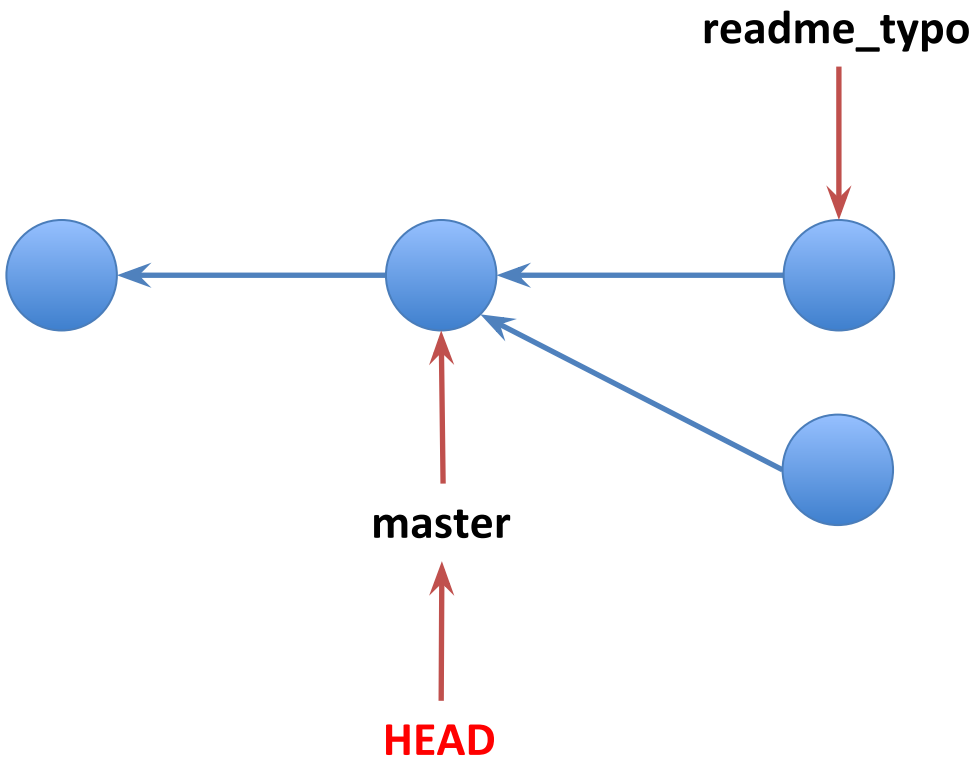

master
readme_typo
HEAD


readme_typo
master
HEAD

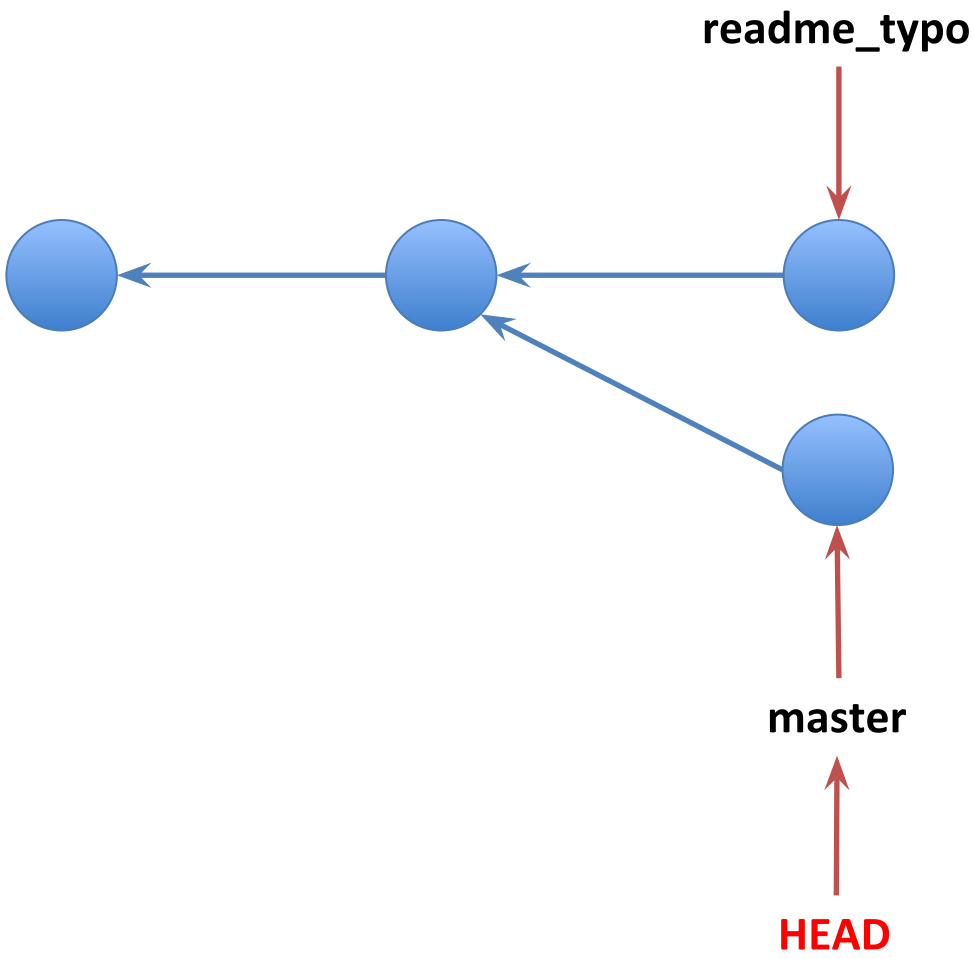
git commit



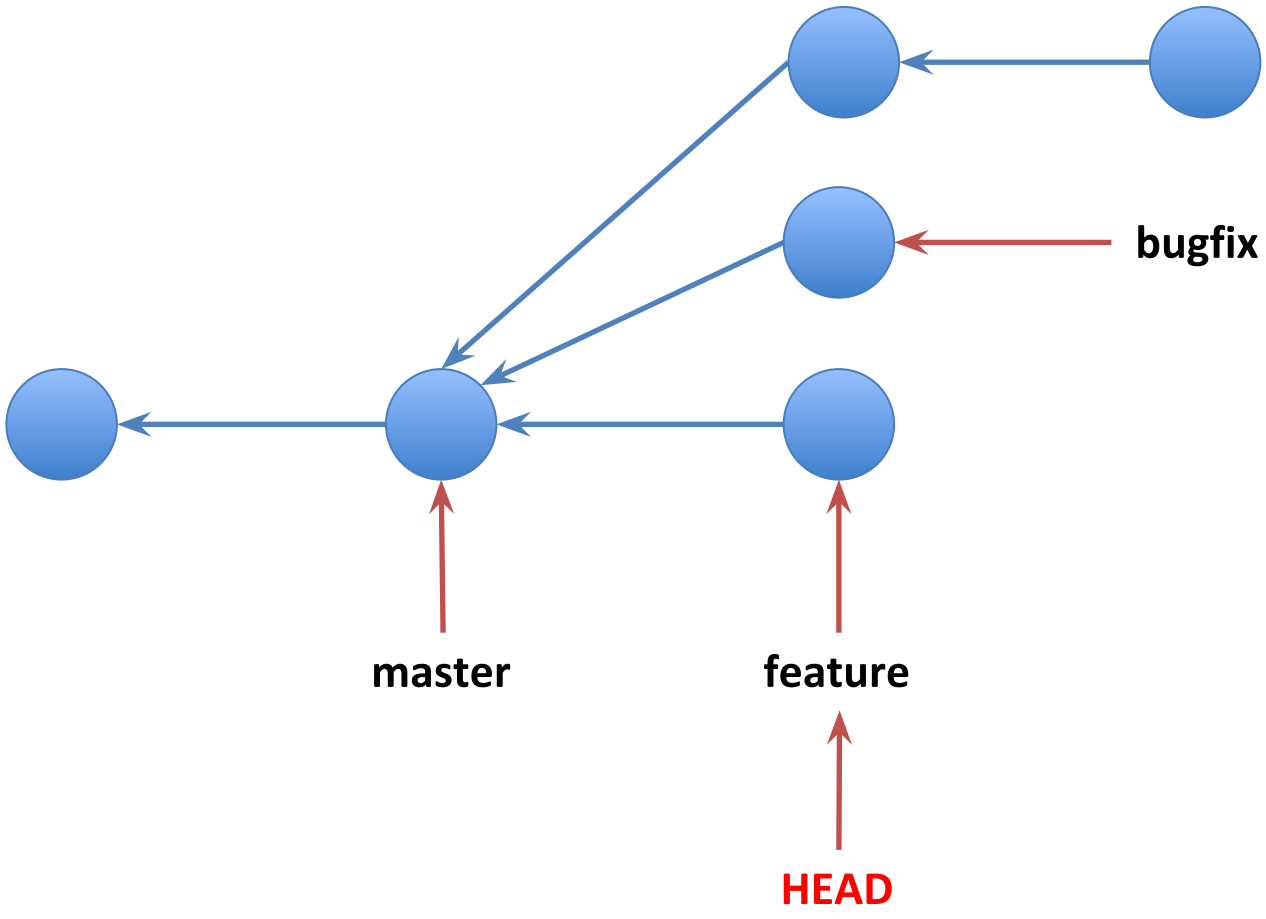
git commit



git commit



Quiz: Δημιούργησε τον γράφο



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

`git merge <branch>`

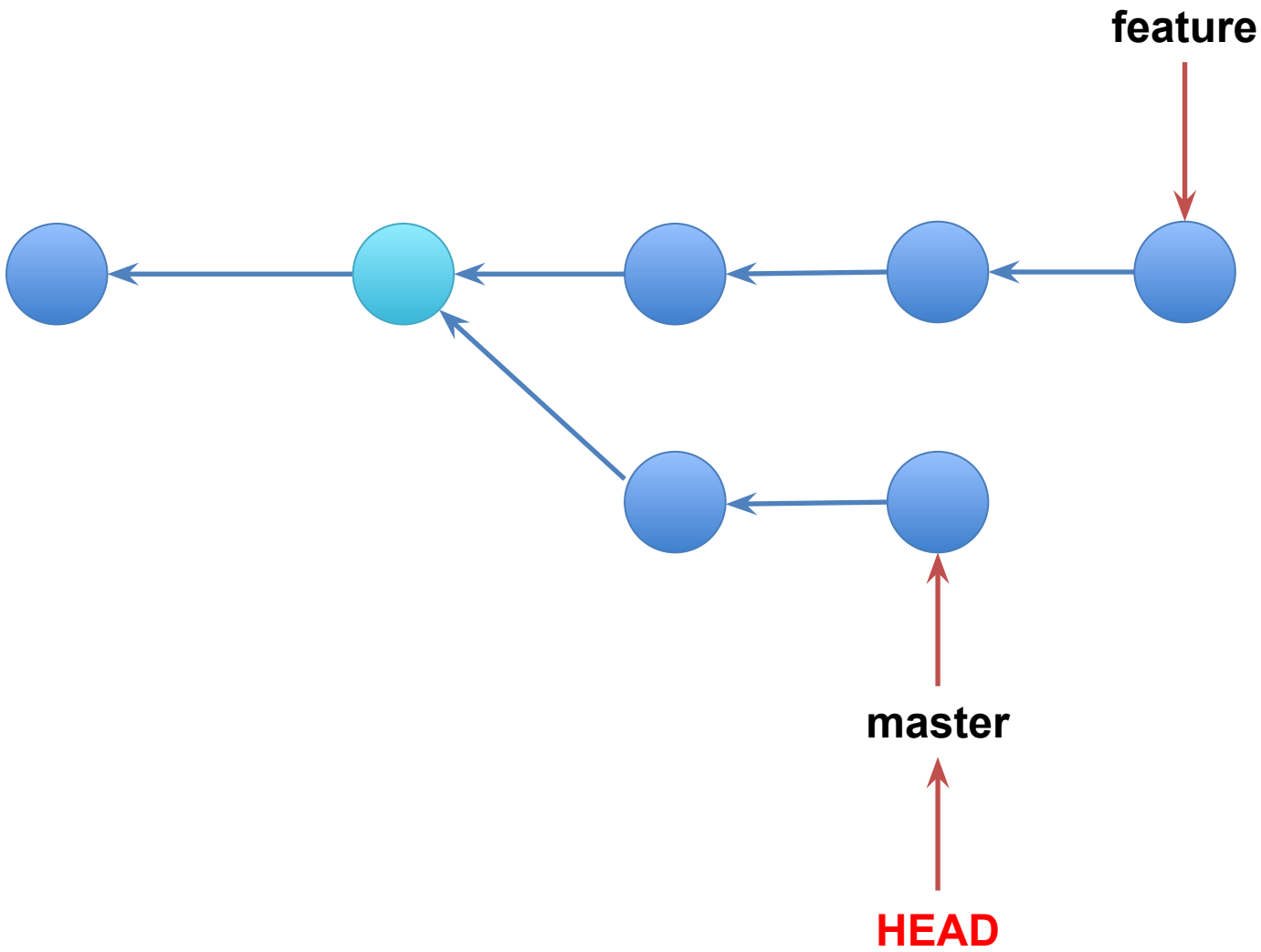
- Ενοποιεί το ιστορικό του branch που περνάς ως παράμετρο με το τωρινό branch
- Προσπαθεί να ενώσει τις αλλαγές στα αρχεία και από τα δύο branches
- Δημιουργεί ένα commit με 2 γονιούς:
  - Το τρέχον branch
  - Το branch που δίνεται ως παράμετρος

Αλγόριθμος merging

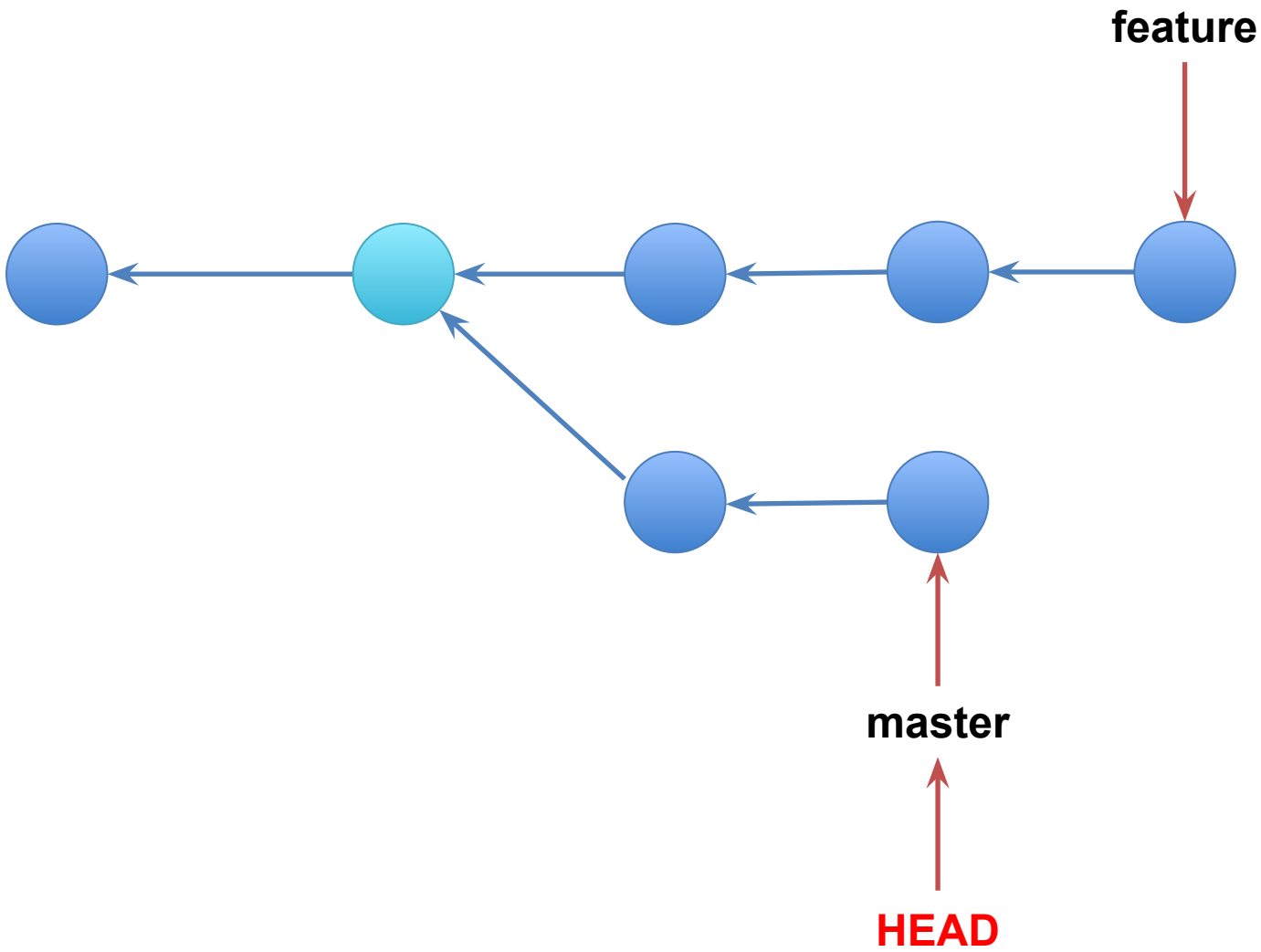
- Έστω ότι βρισκόμαστε στο master
- Τρέχουμε: `git merge feature`
- Αυτό δημιουργεί στο master τις αλλαγές που έγιναν εντωμεταξύ στο feature branch
- Δημιουργεί ένα νέο “merge commit” με δύο γονιούς:
  - master
  - feature

Αλγόριθμος merging

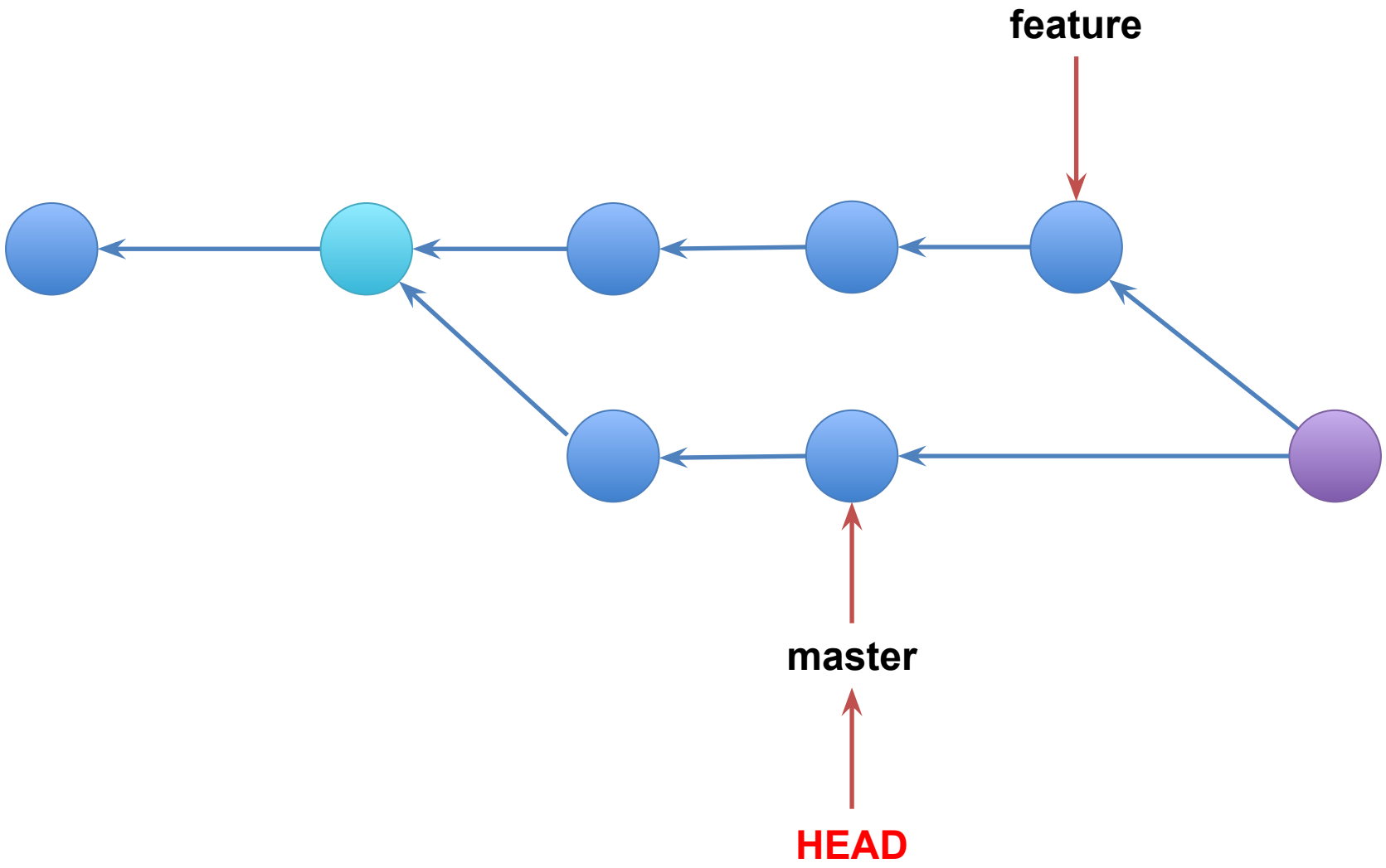
- Εντοπίζεται ο πιο πρόσφατος κοινός πρόγονος ανάμεσα σε master και feature
- Τα commits που πρέπει να εφαρμοστούν για να γίνει το merge είναι όλα τα commits ανάμεσα σε αυτόν τον πρόγονο και το feature branch
- Οι αλλαγές σε αυτά τα commits εφαρμόζονται στο master
- Το master μεταφέρεται στο νέο merge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


feature
master
HEAD

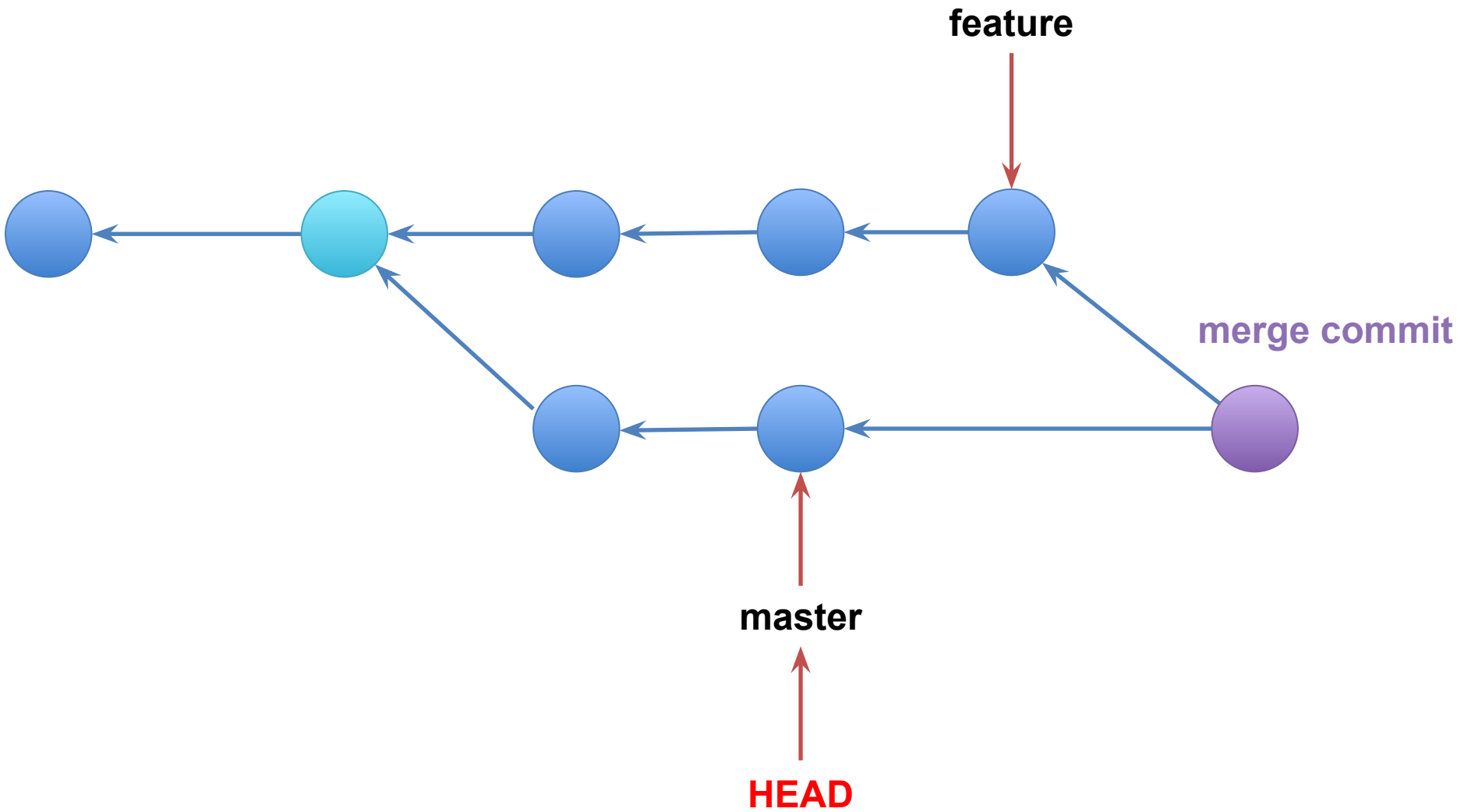
git merge feature



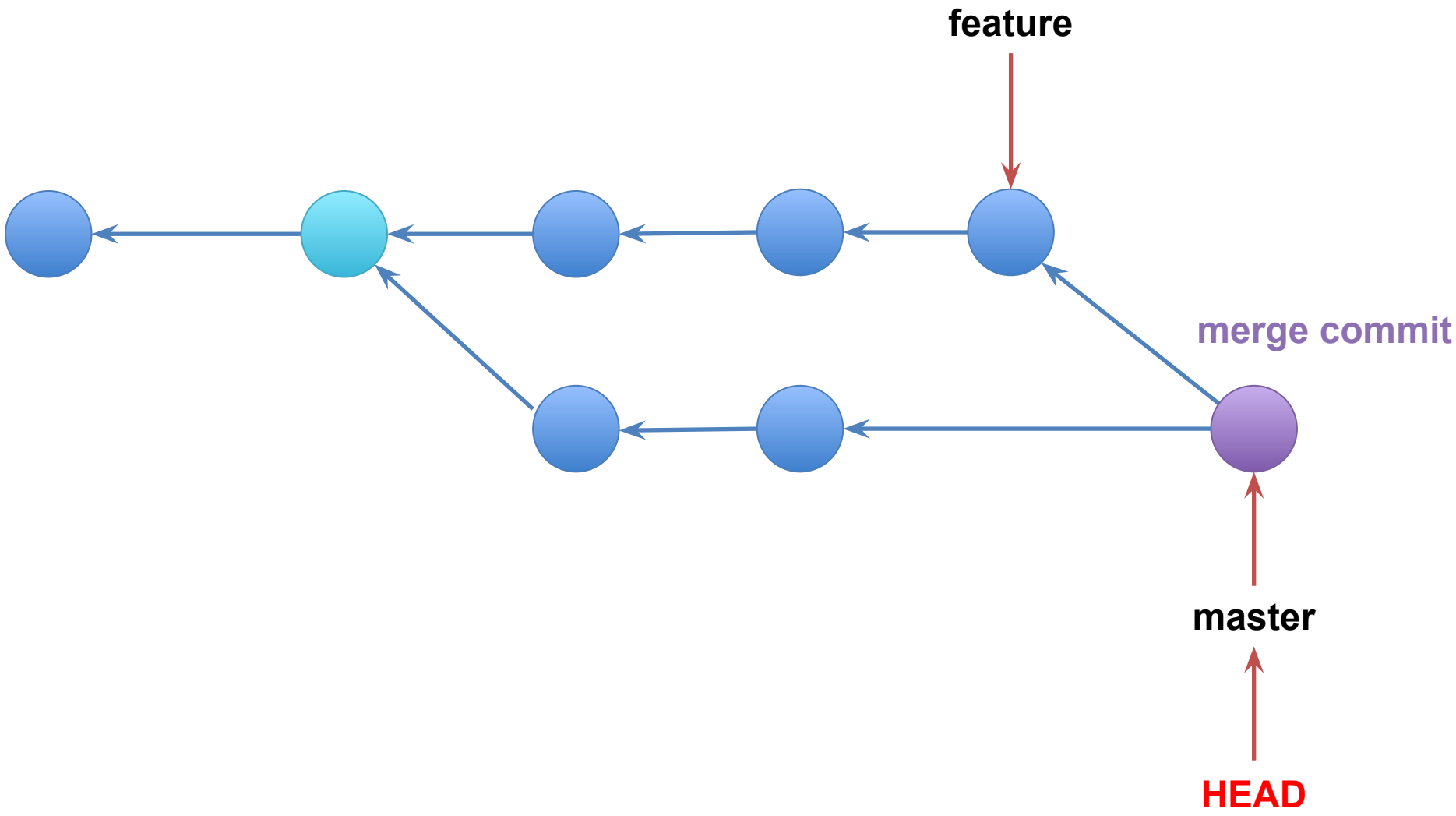
git merge feature



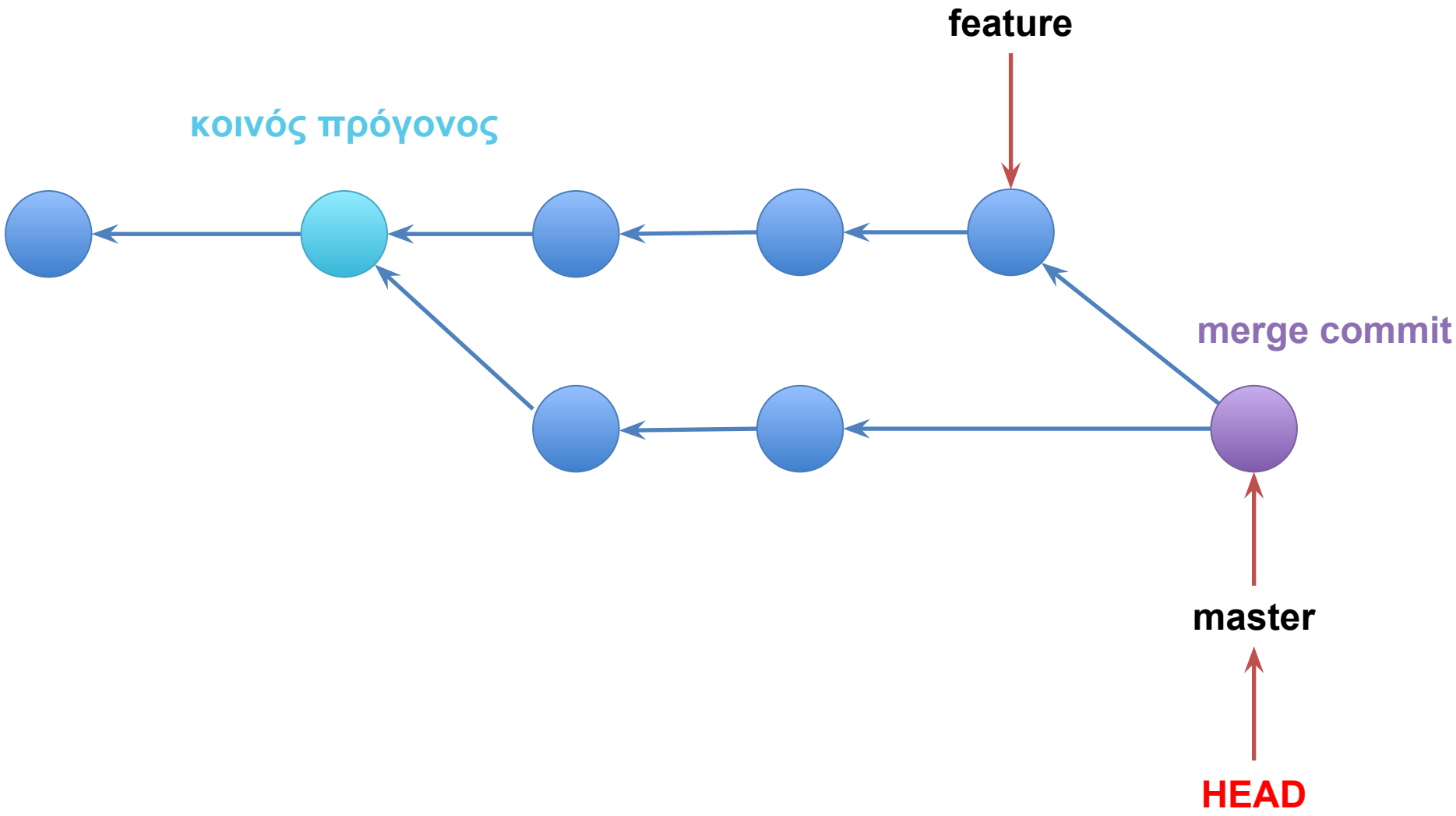
git merge feature



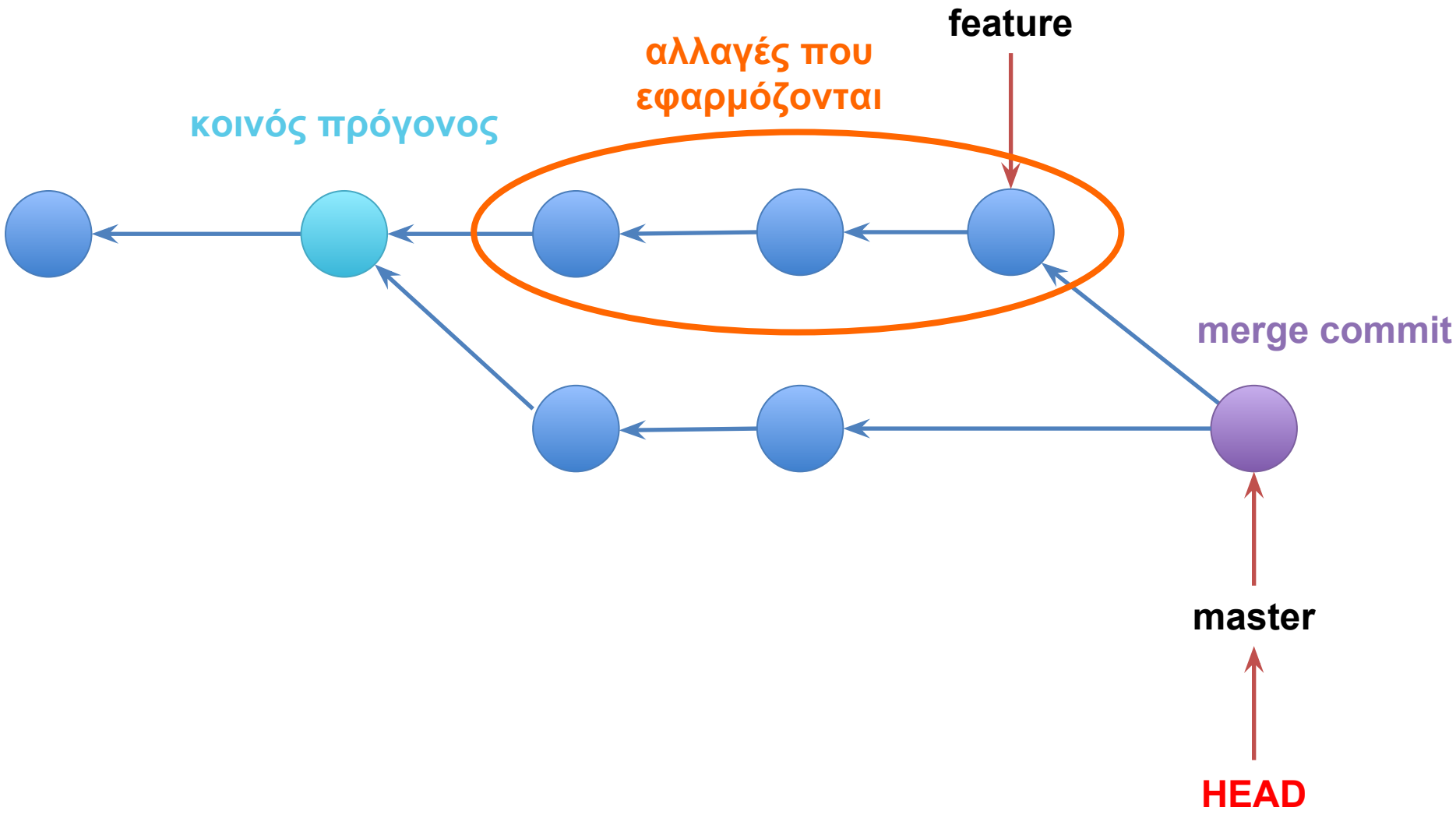
git merge feature



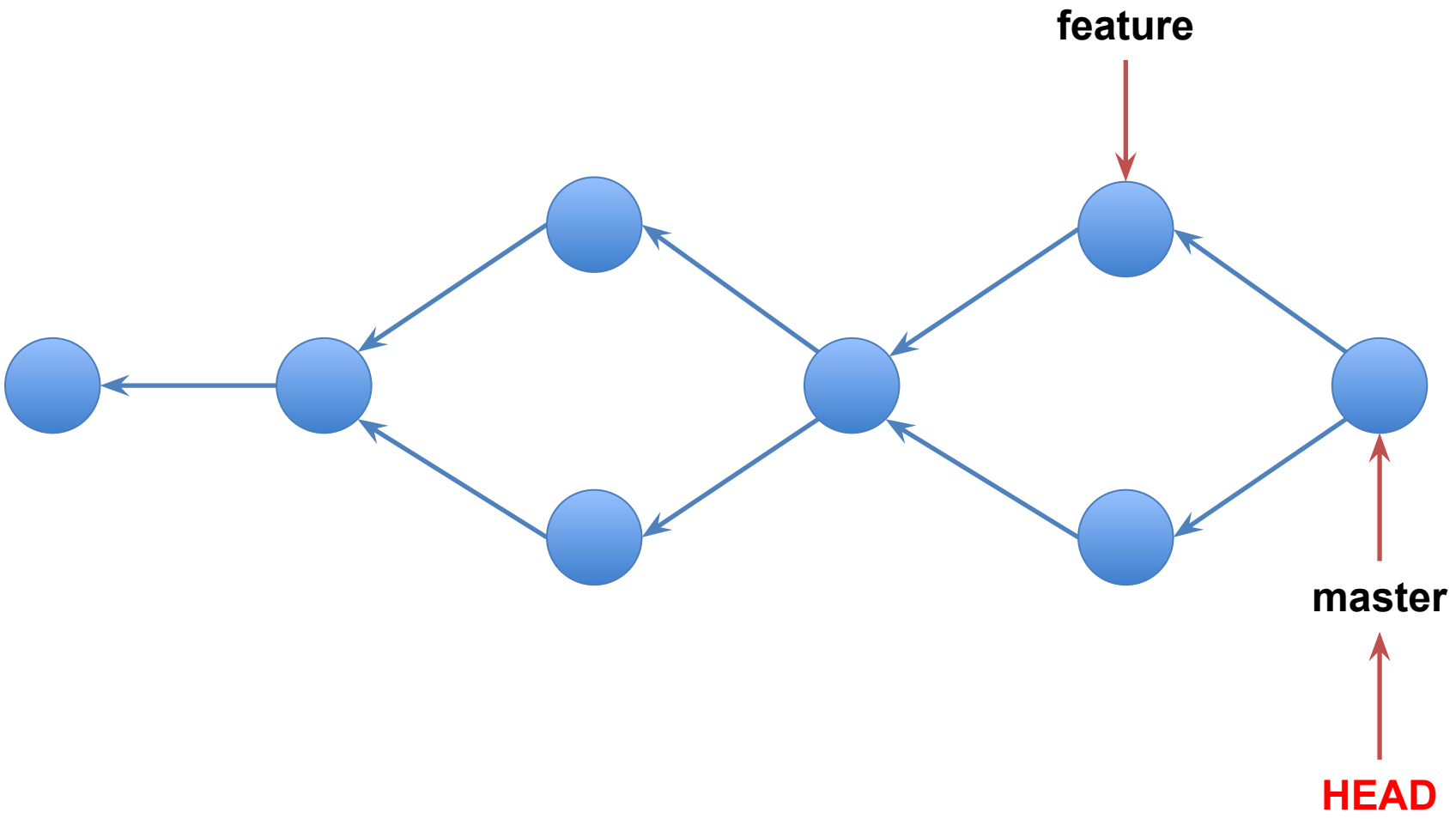
git merge feature



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

- Προτείνουμε να δημιουργείς **ένα branch ανά feature**
- Για **κάθε** μικρό feature (π.χ. αλλαγή χρώματος ενός κουμπιού, διόρθωση ενός bug κλπ.) δημιουργούμε ένα νέο branch
- Κάνουμε τις αλλαγές μας στο νέο branch
- Κάνουμε όσα commits χρειάζονται
- Κάνουμε merge στο master
- Διαγράφουμε το νέο branch

Branching workflow

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

πολλές φορές

Stash

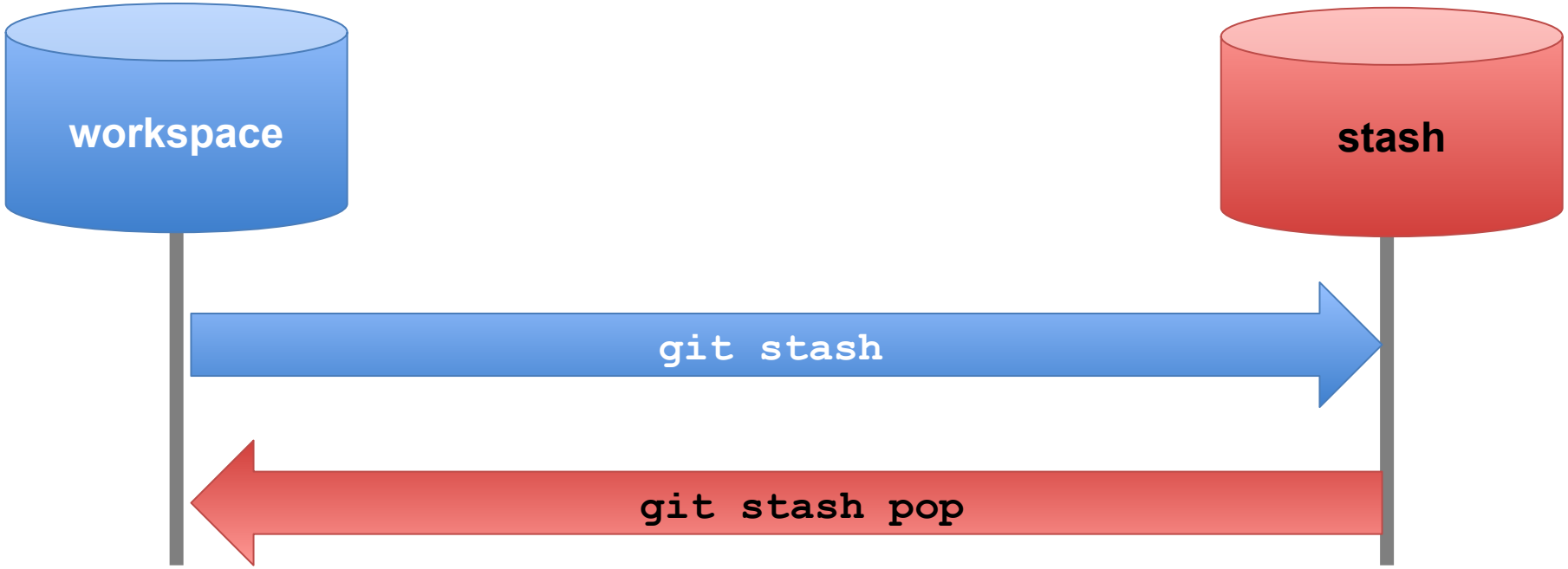
- Μερικές φορές χρειάζεται να αλλάξουμε branch για κάτι έκτακτο (π.χ. hotfix)
- Εντωμεταξύ μπορεί να έχουμε κάνει αλλαγές στο working copy μας
- Επιλογές:
  - commit… όμως δε θέλουμε να κάνουμε commit κάτι τσαπατσούλικο
  - checkout… όμως δε θέλουμε να χάσουμε τις αλλαγές μας

`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

GitHub

Remotes

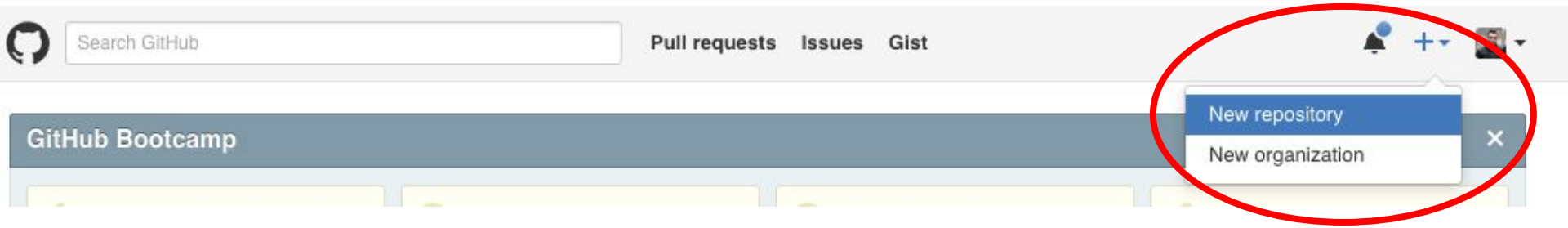
- Ο κώδικας είναι συνεργατικό πράγμα
- Μια ομάδα χρειάζεται πολλά αντίγραφα του κώδικά της, ένα για κάθε προγραμματιστή
- Μερικές φορές χρειάζεται να ανταλλάξουμε κώδικα με αυτά τα αντίγραφα
- Κάθε αντίγραφο με το οποίο ανταλλάσσουμε κώδικα ονομάζεται “remote”

GitHub

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

Φτιάχνοντας το δικό μας repo

- Μπορούμε να φτιάξουμε δικά μας repos μέσω του GitHub



- Απλώς κάνουμε clone το δικό μας repo για να το έχουμε τοπικά

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner **Repository name**

kkanellis / vigilant-guide

Great repository names are short and memorable. Need inspiration? How about **vigilant-guide**.

Description (optional)

Public
Anyone can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

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

Add .gitignore: **None** | Add a license: **None**

Create repository

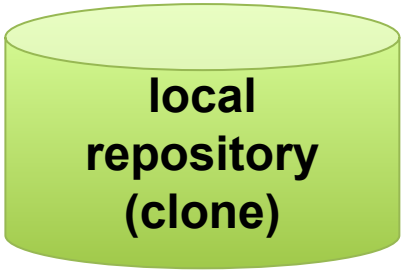
Cloning

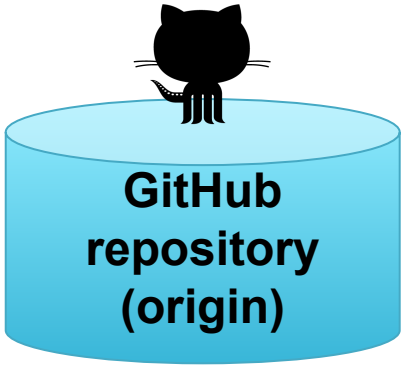
- Τώρα που έχεις ένα δικό σου repo στο GitHub, μπορείς να φτιάξεις ένα αντίγραφο στον υπολογιστή σου

- `git clone https://github.com/`*`your-name`*`/repo-name.git`
- `cd repo-name`

Remotes

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





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 GitHub
- 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
GitHub
Repository

Remote workflow (simple)

```
git checkout master
git pull origin master
vim
git add
git commit
git push origin master
```

πολλές φορές

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”

```
dp@alpha ce421-lab2 (master>) $ git push
To git@github.com:paraschas/ce421-lab2.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'git@github.com:paraschas/ce421-lab2.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
dp@alpha ce421-lab2 (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

Μάθαμε

- Συνεργατική χρήση του git
  - branches
  - remotes
  - push/pull

GitHub Issues

- Bug tracking σύστημα
- Κάθε issue
  - Είναι ένα bug ή ένα task
  - Είναι ανοιχτό (open) ή κλειστό (closed - resolved)
  - Έχει ένα description
  - Έχει comments
- Issues μπορούν να ανοίξουν όλοι (και μη ιδιοκτήτες του repo)

servo / servo

Watch 368 | Unstar 6,340 | Fork 1,047

Code | Issues 1,327 | Pull requests 52 | Wiki | Pulse | Graphs

Filters | is:issue is:open | Labels | Milestones | New issue

1,327 Open | 2,723 Closed | Author | Labels | Milestones | Assignee | Sort

- **Publish the CSS support list during doc builds** A-build E-easy L-python — 1
  #10196 opened 3 hours ago by larsbergstrom
- **Callback code does weird things with compartments** — 1
  #10192 opened 8 hours ago by Ms2ger
- **Rename style struct types to eliminate the T** A-content/css E-easy I-cleanup — 1
  #10185 opened 22 hours ago by bholley
- **Remove backspace-to-navigate** C-assigned E-easy — 0
  #10184 opened 22 hours ago by Manishearth
- **forward navigation key only works with shift** E-easy — 1
  #10183 opened 22 hours ago by matthiaskrgr
- **Implement *-reverse flex-directions** A-layout/flex — 0
  #10181 opened a day ago by danlrobertson
- **Add a build configuration to allow profiling heap allocations from Instruments.app** A-build I-perf-bloat — 0
  #10180 opened a day ago by jdm
- **Intermittent timeout in /websockets/constructor/016.html** I-intermittent — 0
  #10177 opened a day ago by jdm
- **Create an HTTP global state object** A-network C-assigned E-easy I-refactor — 5
  #10175 opened a day ago by jdm
- **Annotate test failures with bugs** — 1
  #10172 opened a day ago by Ms2ger

servo / servo

Watch 368 | Unstar 6,340 | Fork 1,047

Code | Issues 1,327 | Pull requests 52 | Wiki | Pulse | Graphs

Publish the CSS support list during doc builds #10196

New issue

Open **larsbergstrom** opened this issue 3 hours ago · 1 comment

larsbergstrom commented 3 hours ago

In #9333, a script was added that prints out all of the properties support by Servo as JSON. In this project, we would edit the file https://github.com/servo/servo/blob/master/etc/ci/upload_docs.sh to run that script (via `python components/style/list_properties.py`) and store the output as a JSON file in the documentation tree so that it is uploaded after each build lands in master.

For extra brownie bonus points, adding a HTML file with some nice CSS styles to display that list to the https://github.com/servo/servo/tree/master/etc/doc.servo.org directory would be a really nice bonus! We could then link to that directly from servo.org.

This would fully close #3954.

larsbergstrom added E-easy L-python A-build labels 3 hours ago

larsbergstrom commented 3 hours ago

cc **@SimonSapin**

Labels

A-build
E-easy
L-python

Milestone

No milestone

Assignee

No one assigned

Notifications

Subscribe

You're not receiving notifications from this thread.

1 participant

Αναγνωριστικά GitHub Issues

- Μοναδικό αναγνωριστικό για κάθε issue μέσα στο repo
- Μπορούμε να αναφερθούμε σε αυτό
  - από το commit message:
    - `git commit -m "Refactor code for issue #765"`
    - `git commit -m "Closes #765"` (θα κλείσει το issue 765)
  - από PR descriptions παρομοίως
- Το GitHub μετατρέπει τις αναφορές σε links

`git checkout <commit>`

- Επαναφέρει το working copy μας στην κατάσταση που ήταν το commit που επιλέξαμε
- Αλλάζει το HEAD ώστε να δείχνει σε αυτό το commit
  - Το HEAD πλέον δεν δείχνει σε κάποιο branch αλλά σε κάποιο commit απευθείας
  - Οπότε είμαστε σε state detached HEAD

`git checkout <commit>`

- Δεν μπορούμε να κάνουμε commits σε αυτή τη κατάσταση
- Πρέπει να κάνουμε πάλι checkout σε κάποιο branch
- Χρησιμοποιείται κυρίως για να δούμε πώς ήταν ο κώδικας κάποια στιγμή

GitHub show

- Μας δείχνει τα ίδια πράγματα με το git show αλλά με πιο ευανάγνωστο τρόπο

[hw2] prob3: bugfix

master

Browse files

kkanellis committed on 7 Nov 2015 — 1 parent 28fbadf — commit abf9bcc199d750ccfa601610a81a06be27e60932

Showing **1 changed file** with **2 additions** and **2 deletions**.

Unified | Split

4 homework2/prob3/cars.c — View

```diff
@@ -84,8 +84,8 @@ void * red_car (void * args)
84          red_waiting--; red_crossing++;
85          pthread_mutex_unlock(&red_queue);
86      }
87 -    pthread_mutex_unlock(&mtx);
88  }
89
90  log_info("[%d] RED: Exited bridge", car_id);
91  return NULL;
@@ -137,8 +137,8 @@ void * blue_car (void *args)
137         blue_waiting--; blue_crossing++;
138         pthread_mutex_unlock(&blue_queue);
139     }
140 -   pthread_mutex_unlock(&mtx);
141 }
142
143 log_info("[%d] BLUE: Exited bridge", car_id);
144 return NULL;
```

```diff
84          red_waiting--; red_crossing++;
85          pthread_mutex_unlock(&red_queue);
86      }
87  }
88 +pthread_mutex_unlock(&mtx);
89
90  log_info("[%d] RED: Exited bridge", car_id);
91  return NULL;
137         blue_waiting--; blue_crossing++;
138         pthread_mutex_unlock(&blue_queue);
139     }
140 }
141 +pthread_mutex_unlock(&mtx);
142
143 log_info("[%d] BLUE: Exited bridge", car_id);
144 return NULL;
```

Conflicts

- Όταν κάνουμε merge, μπορεί να μην μπορεί να γίνει αυτόματα, π.χ.
  - ένας προγραμματιστής άλλαξε ένα αρχείο, ενώ ένας άλλος το διέγραψε
  - δύο άτομα άλλαξαν το ίδιο σημείο ενός αρχείου (διόρθωσαν το ίδιο bug με διαφορετικό τρόπο)
- Το git δεν ξέρει ποια είναι η σωστή αλλαγή
- Τότε μας λέει ότι υπάρχουν “conflicts” (διενέξεις)

```
sfi@atlas:~/git-example *$ git add counter.c
sfi@atlas:~/git-example *$ git commit -m "implement counter to 10"
[master (root-commit) c06810f] implement counter to 10
 1 file changed, 12 insertions(+)
 create mode 100644 counter.c
sfi@atlas:~/git-example (master)$ git checkout -b feature
Switched to a new branch 'feature'
sfi@atlas:~/git-example (feature)$ vim counter.c
sfi@atlas:~/git-example (feature)*$ git add counter.c
sfi@atlas:~/git-example (feature)*$ git commit
[feature 902f690] add times variable
 1 file changed, 2 insertions(+), 1 deletion(-)
sfi@atlas:~/git-example (feature)$ git checkout master
Switched to branch 'master'
sfi@atlas:~/git-example (master)$ vim counter.c
sfi@atlas:~/git-example (master)*$ git add counter.c
sfi@atlas:~/git-example (master)*$ git commit
[master 8839bfd] add counter variable
 1 file changed, 2 insertions(+), 1 deletion(-)
sfi@atlas:~/git-example (master)$ git merge feature
Auto-merging counter.c
CONFLICT (content): Merge conflict in counter.c
Automatic merge failed; fix conflicts and then commit the result.
sfi@atlas:~/git-example (master)*$
```

Επίλυση conflicts

- Το git έχει αλλάξει το αρχείο μας
  - Ενοποιεί όλες τις αλλαγές που μπορεί αυτόματα
  - Τις υπόλοιπες τις αφήνει σ’ εμάς
- Ανοίγουμε το αρχείο που έχει conflicts με τον editor μας
- Ψάχνουμε για <<<<< ===== >>>>>

counter.c

```
1   #include <stdio.h>
2
3   int main(int argc, char * argv[]) {
4       int i;
5
6   <<<<<<< HEAD
7       int counter = 10;
8       for (i = 1; i <= counter; i++) {
9   =======
10      int times = 10;
11      for (i = 1; i <= times; i++) {
12  >>>>>>> feature
13          printf("%d\n", i);
14      }
15
16      return 0;
17  }
18
```

Έκδοση στο ***master***

Έκδοση στο ***feature***

Επίλυση conflicts

- Αντικαθιστούμε το τμήμα με το σωστό κώδικα
  - σε όλα τα τμήματα και αρχεία που έχουν conflicts
- Κάνουμε git add κάθε αρχείο
- **Προσοχή**: Δεν κάνουμε άλλες αλλαγές στα αρχεία πέρα από την επίλυση conflicts
- Κάνουμε git commit
  - Έτσι δημιουργείται το merge commit
  - Ο τελικός γράφος είναι ο ίδιος σαν να μην είχαμε conflicts

counter.c

```c
#include <stdio.h>

int main(int argc, char * argv[]) {
    int i;

    int times = 10;
    for (i = 1; i <= times; i++) {
        printf("%d\n", i);
    }

    return 0;
}
```

```
sfi@atlas:~/git-example (master)*$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   counter.c

no changes added to commit (use "git add" and/or "git commit -a")
sfi@atlas:~/git-example (master)*$ git add counter.c
sfi@atlas:~/git-example (master)*$ git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

        modified:   counter.c

sfi@atlas:~/git-example (master)*$ git commit -m "keep 'times' variable"
[master 1033707] keep 'times' variable
sfi@atlas:~/git-example (master)$ git status
On branch master
nothing to commit, working directory clean
sfi@atlas:~/git-example (master)$
```

Μάθαμε

- Περισσότερα για το GitHub
  - issues
  - show
- Git merging
  - + επίλυση conflicts

Αφήσαμε εκτός

- Προχωρημένες τεχνικές git
  - Undo (revert, reset)
  - Blame, Tag, Revert
  - Cherry pick, Rebase
- Workflows και συνεργατικές τεχνικές

Extras

- Get the repo status in the command prompt
- Use SSH for password-less GitHub operations
- Try Git: short online tutorial
- Learn Git Branching
- Official and free Git book

Ευχαριστούμε!