Git 101

Τι είναι ένα version control system?

Είναι ένα σύστημα αναθεώρησης κώδικα που διευκολύνει το προγραμματιστή να **διαχειρίζεται τις αλλαγές στα αρχεία** του και να **συνεργάζεται με άλλους προγραμματιστές**.

Το Git προσφέρει

- Κατανεμημένη Ανάπτυξη
- Ευκολία για μη γραμμική ανάπτυξη (merging/branching)
- Ευκολία για Ανάπτυξη μεγάλων projects
- Ιστορικό Ανάπτυξης
- Unix philosophy of software design

Projects που αναπτύσονται με Git

- Git (!)
- Linux Kernel
- Perl
- Gnome
- QT
- Ruby on Rails
- Android
- PostgreSQL
- more...

Repository

Το git αποθηκεύει τις πληροφορίες σε μία δομή δεδομένων που ονομάζεται **repository**.

Ένα repo αποτελείται από:

- Ένα σύνολο από **commits**
- Ένα σύνολο από αναφορές στα commits, τα **heads**

Repository

Το repository αποθηκεύεται τοπικά στο ίδιο directory με το project, σε ένα υποφάκελο με το όνομα **.git** .

Commit

- Ένα σύνολο αρχείων που αντιπροσωπεύουν τη κατάσταση του project μία δεδομένη χρονική στιγμή
- Αναφορές στα *πατρικά* commits
- Ένα SHA-1 όνομα (40-character string) που το προσδιορίζει μοναδικά.

Head

Ένα head είναι μία αναφορά σε ένα commit.

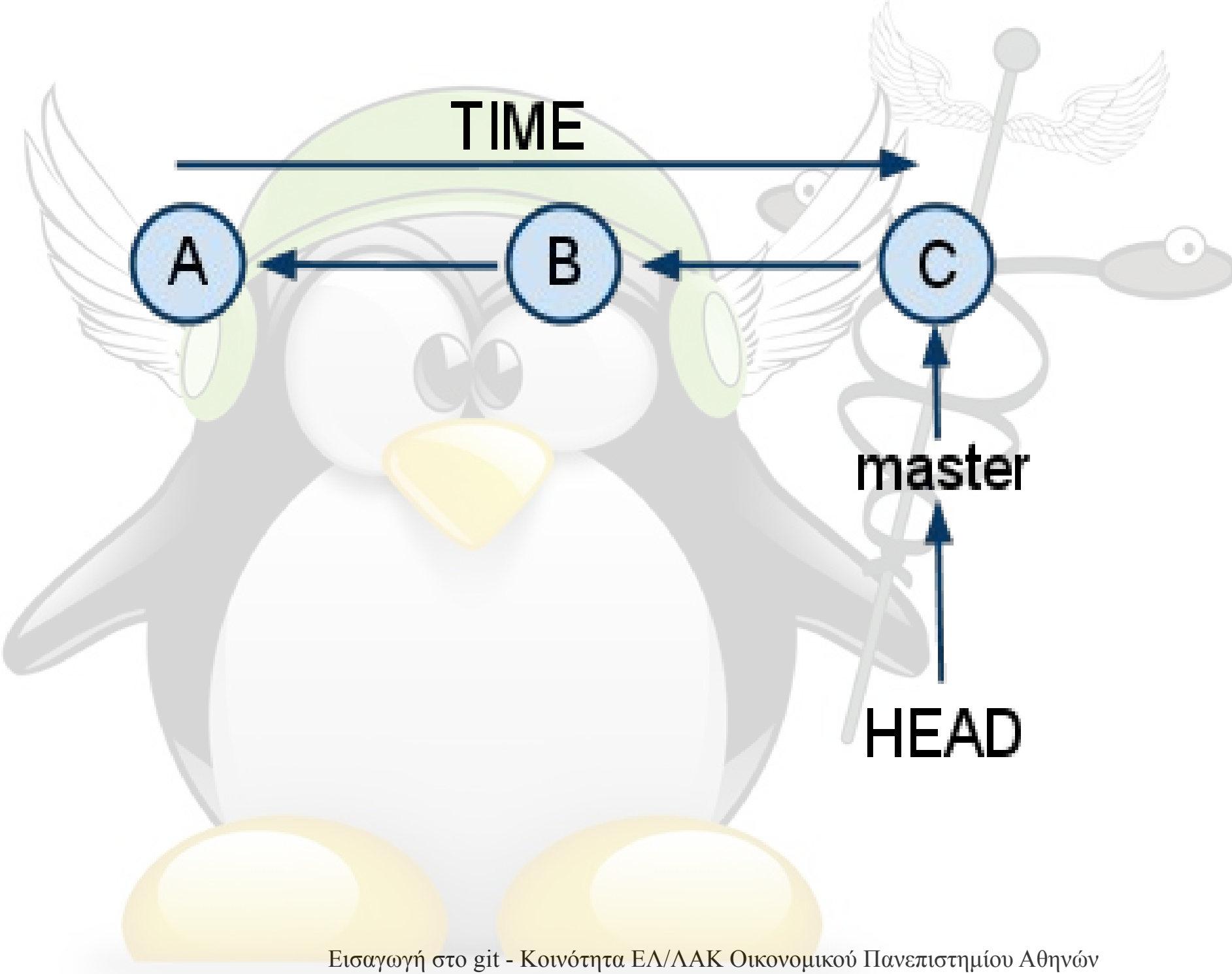
/* current head → HEAD */

Δημιουργία ενός απλού repository

mkdir [project]
cd [project]
git init

Το πρώτο μας commit :)

- Πρέπει να πούμε στο Git ποιά αρχεία θα συμπεριλάβει
  git add
- Να καλέσουμε το
  git commit που θα φτιάξει το commit!


TIME
A
B
C
master
HEAD

Άλλες χρήσιμες εντολές

- ***git log*** **–** δείχνει το ιστορικό των commits ξεκινώντας από το HEAD ως το αρχικό commit
- ***git status*** – δείνχνει ποιά αρχεία άλλαξαν από το HEAD ως τη τωρινή κατάσταση του project
- ***git diff*** – δείχνει τις διαφορές μεταξύ του HEAD και της τωρινής κατάστασης του project
- ***git [mv|rm]*** – επιλέγει αρχεία για μεταφορά/μετανομασία και αφαίρεση αντίστοιχα

Αναφορά σε commit

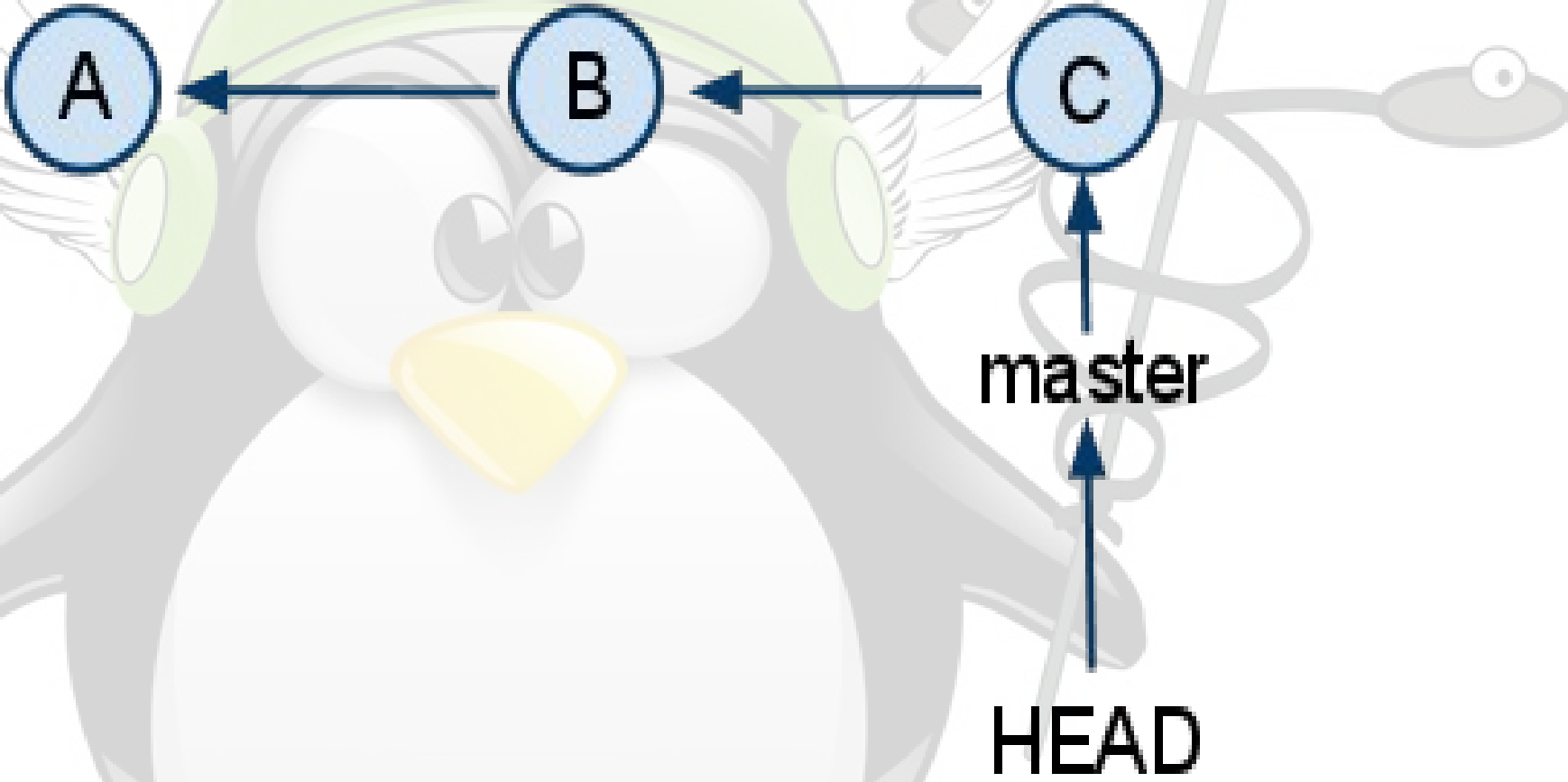
- Με το SHA1 όνομα (φαίνεται στο **git log**) ή τους πρώτους χαρακτήρες από το SHA1 όνομα (~6)
- Με κάποιο head που έχουμε ορίσει. Πχ Με το HEAD αναφερόμαστε στο τελευταίο commit
- Σχετικά με ένα άλλο commit – ένα ^ μετά το commit αναφέρεται στο parent commit.
  π.χ. HEAD^ είναι ο πατέρας αυτού του commit

Branching

Δημιουργία νέων κλαδιών commits ώστε να μην επιρεάσουμε το κεντρικό.

branch ≈ head

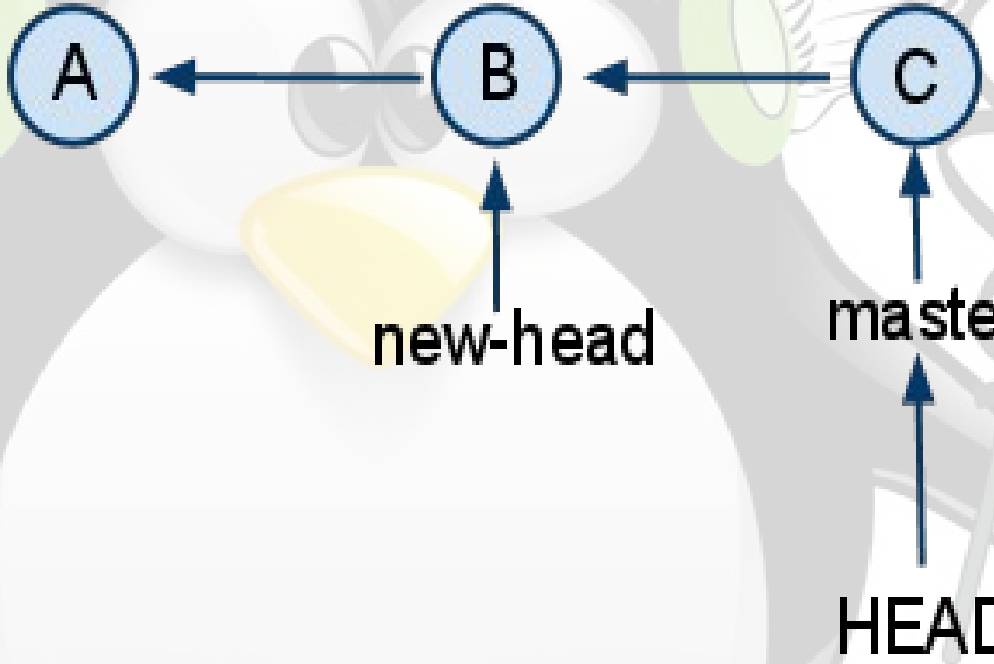
- branch: αναφέρεται σε ένα head και όλα τα προηγούμενα commits
- head: αναφέρεται στο πιο πρόσφατο commit ενός branch


A
B
C
master
HEAD

Δημιουργία Branch

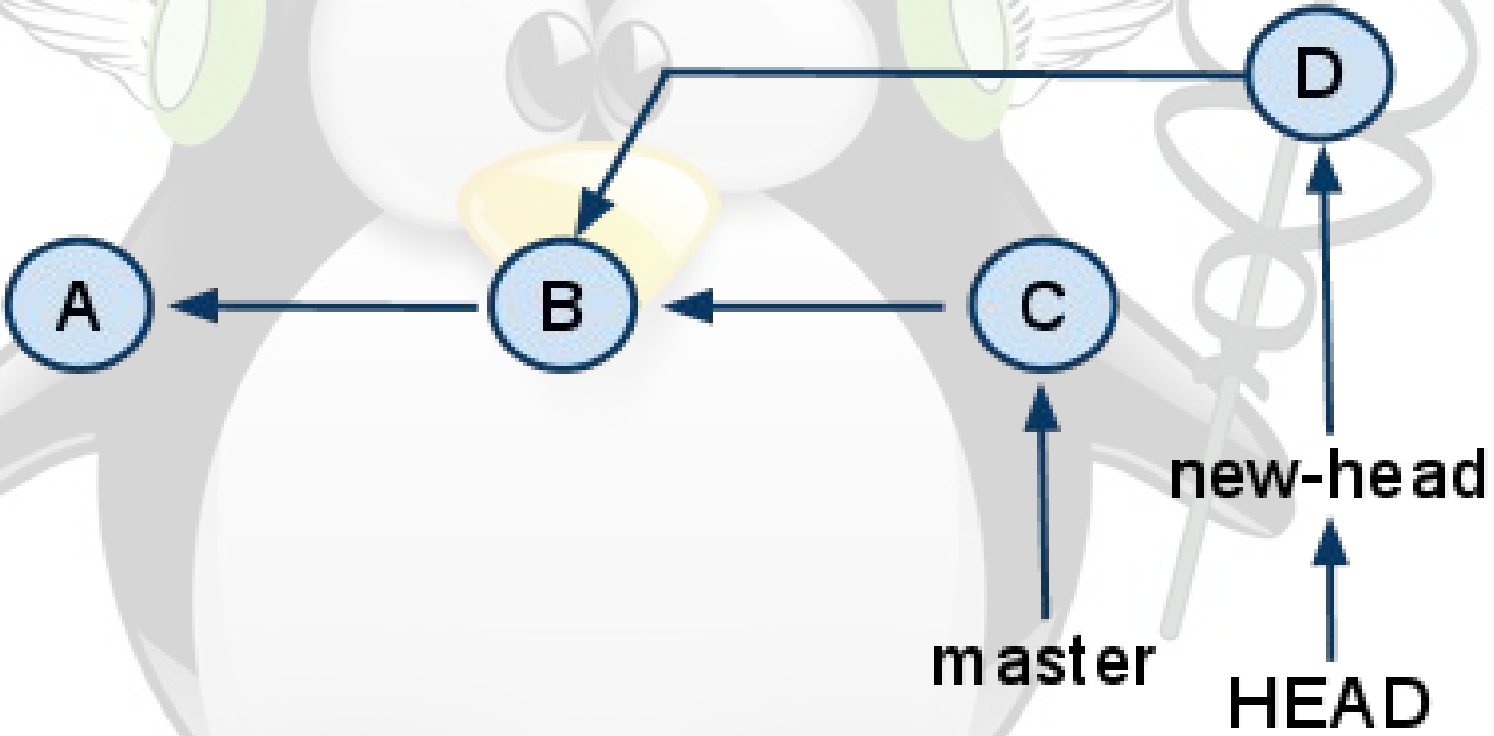
- Με την εντολή ***git log*** βρίσκουμε το SHA1 όνομα του commit που θέλουμε να δουλέψουμε (έστω B)
- ***git branch [new-head-name] [reference-to-B]***

π.χ. *git branch new-head HEAD^*



Αλλάζοντας branches

git checkout [head-name]



Και άλλες χρήσιμες εντολές!

- ***git branch*** χωρίς ορίσματα δείχνει τα υπάρχοντα heads με ένα * στο HEAD
- ***git diff [head1]..[head2]*** δείχνει τη διαφορά ανάμεσα στα commits του head2 και head1
- ***git diff [head1]...[head2]*** δείχνει τη διαφορά ανάμεσα στα commits του head2 και του κοινού προγόνου του με το head1
- ***git log [head1]..[head2]*** δείχνει το ιστορικό αλλαγών για το head2 και του κοινού προγόνου μετο head1.

Συνήθης χρήση Branching

- Ένα main branch σε releaseable state και άλλα branches για τη δημιουργία νέων features.
- Κάθε developer δουλεύει τα features του στο δικό του branch για να μη χαλάει τη δουλειά των άλλων και να παραμένει το project σε releaseable state.

Merging

Αφού αναπτύξουμε τα νέα features στα ξεχωριστά branches πρέπει κάπως να τα εισάγουμε στο master branch!

git merge [head]

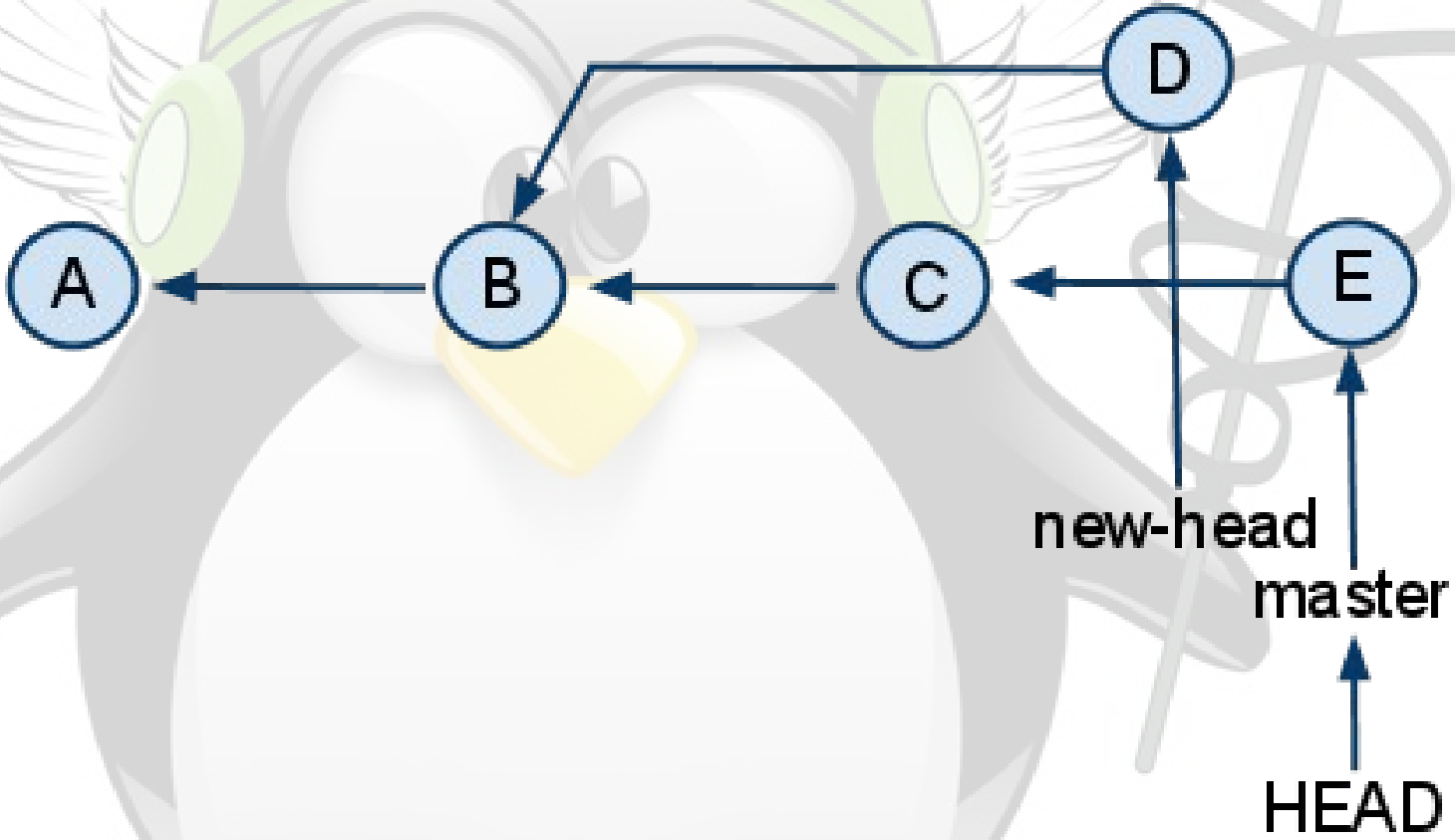
Τι κάνει το merge

- Εντοπίζει το κοινό πρόγονο του HEAD και του head που θα γίνει merge (έστω *merge*)
- Αν ο πρόγονος == *merge* τότε δε κάνει τίποτα.
  Αν ο πρόγονος == HEAD τότε **fast forward merge**
- Διαφορετικά εντοπίζει διαφορές μεταξύ πρόγονου και merge.
- Δοκιμάζει να τα συγχωνεύσει σε ένα αρχείο.
- Αν δεν υπάρχει σύγκρουση δημιουργεί νέο commit. Το HEAD δείχνει σε αυτό (δλδ κάνει checkout).
- Αν υπάρχει σύγκρουση δείχνει που υπάρχει πρόβλημα και ενημερώνει το χρήστη.

Τι κάνει λοιπόν

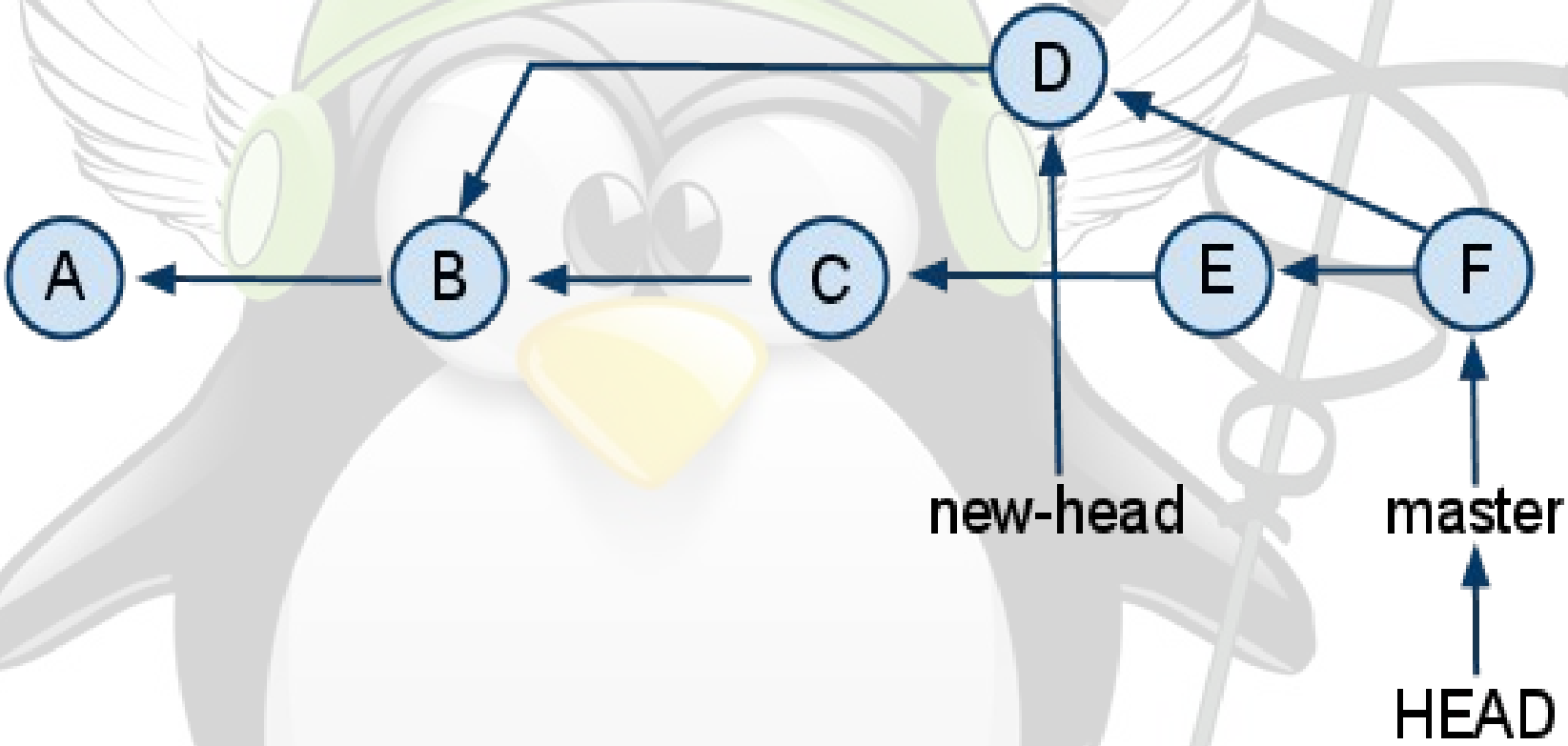
Ας το δούμε !

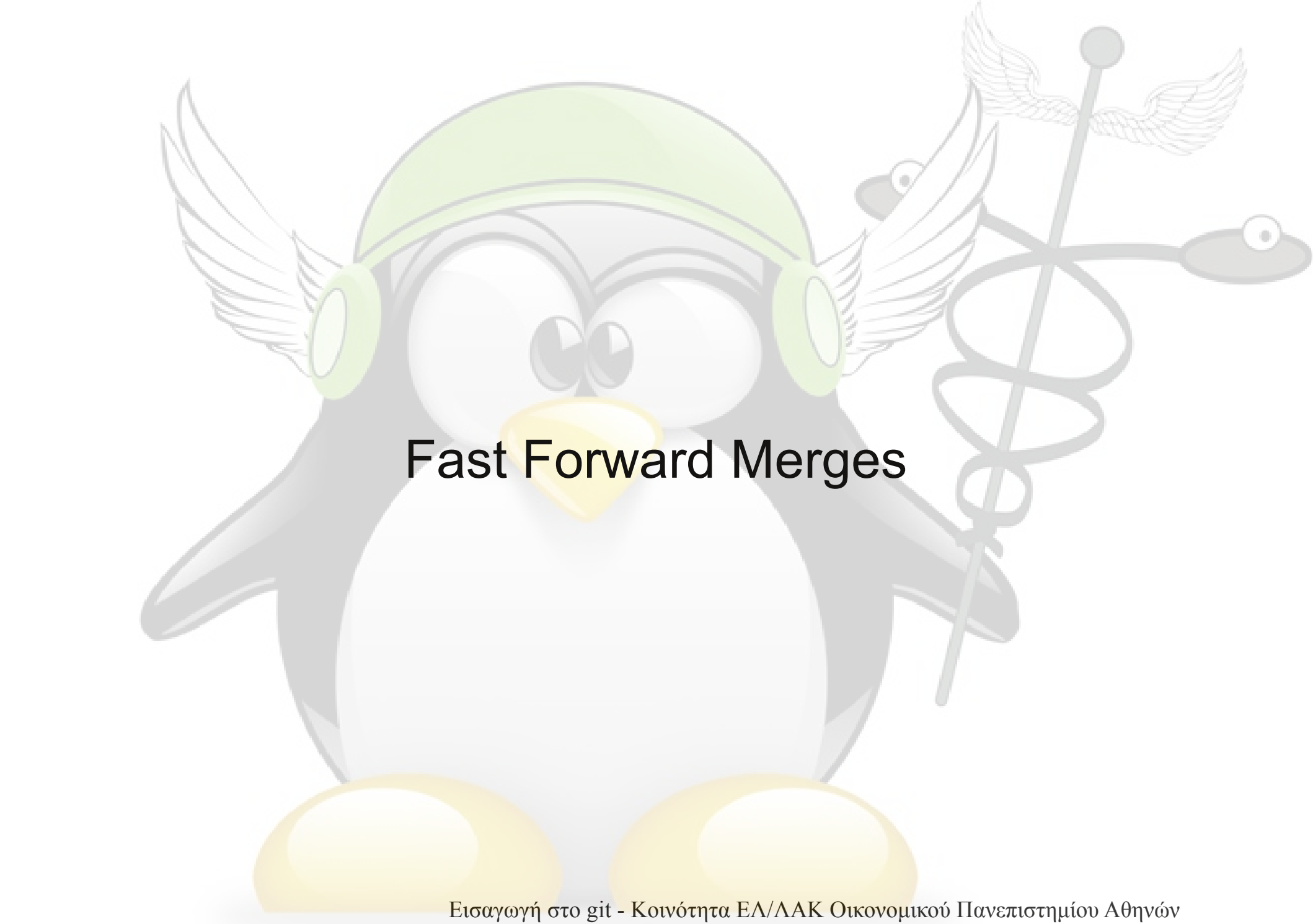
Έστω ότι έχουμε αυτό το δέντρο



git merge new-head

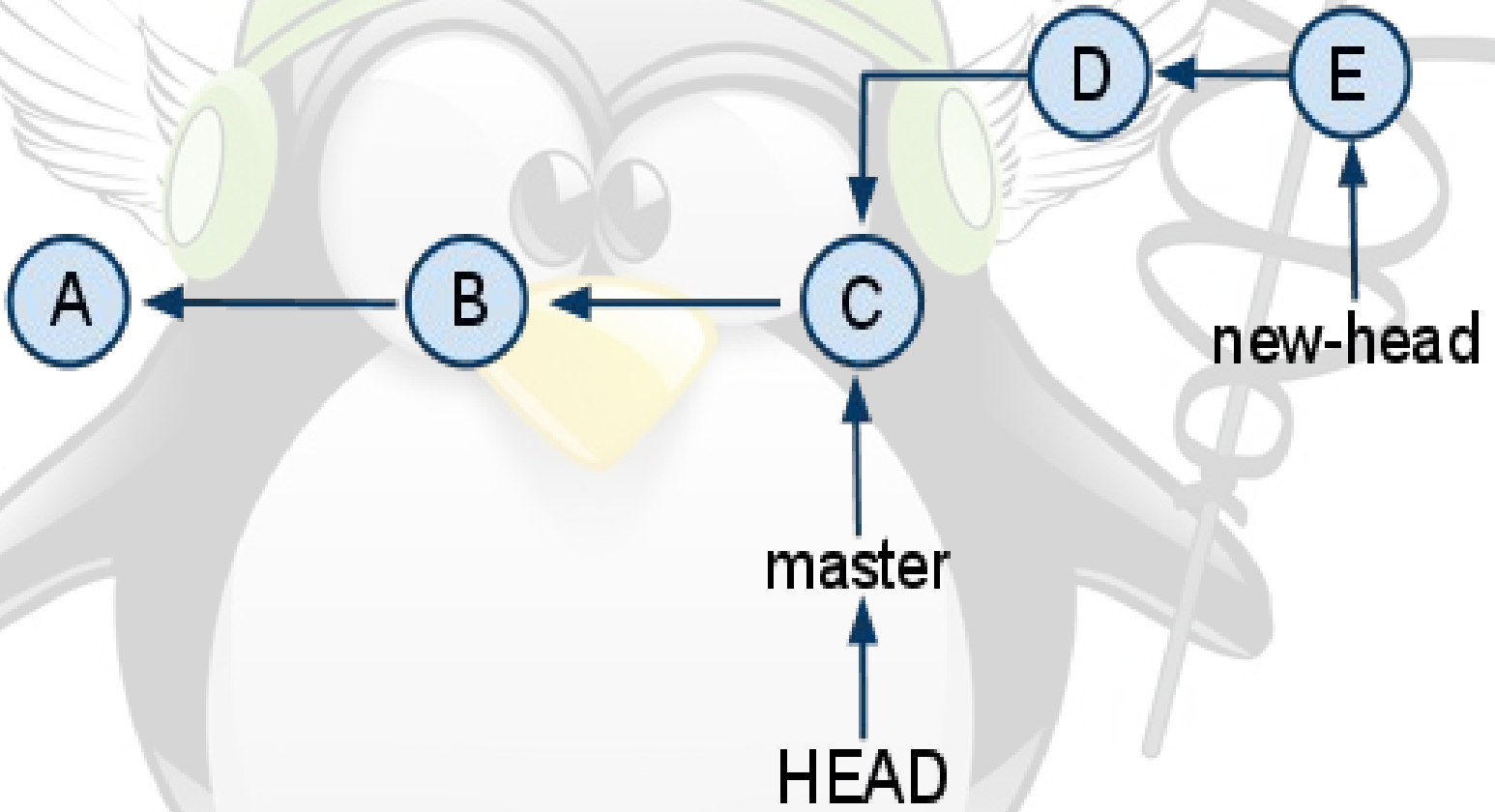
Αν δεν υπάρχει σύγκρουση



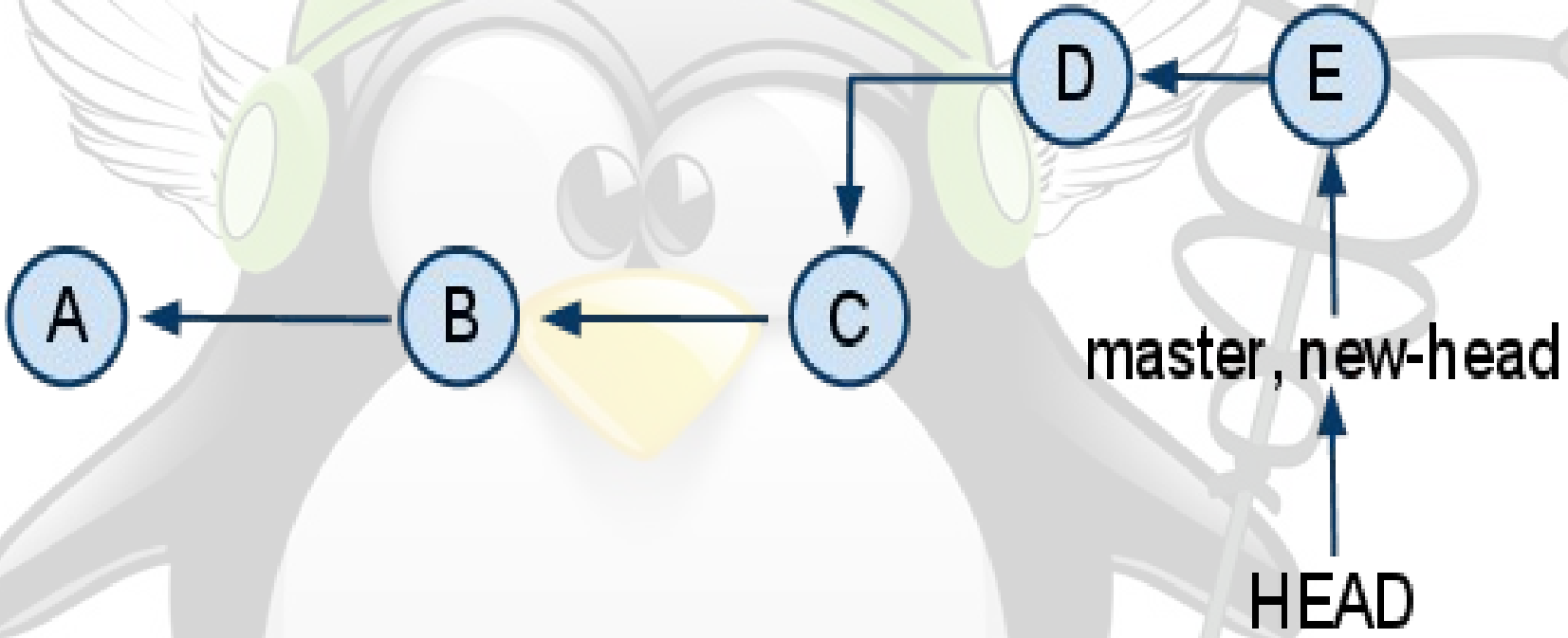


Fast Forward Merges

Έστω



Και γίνεται

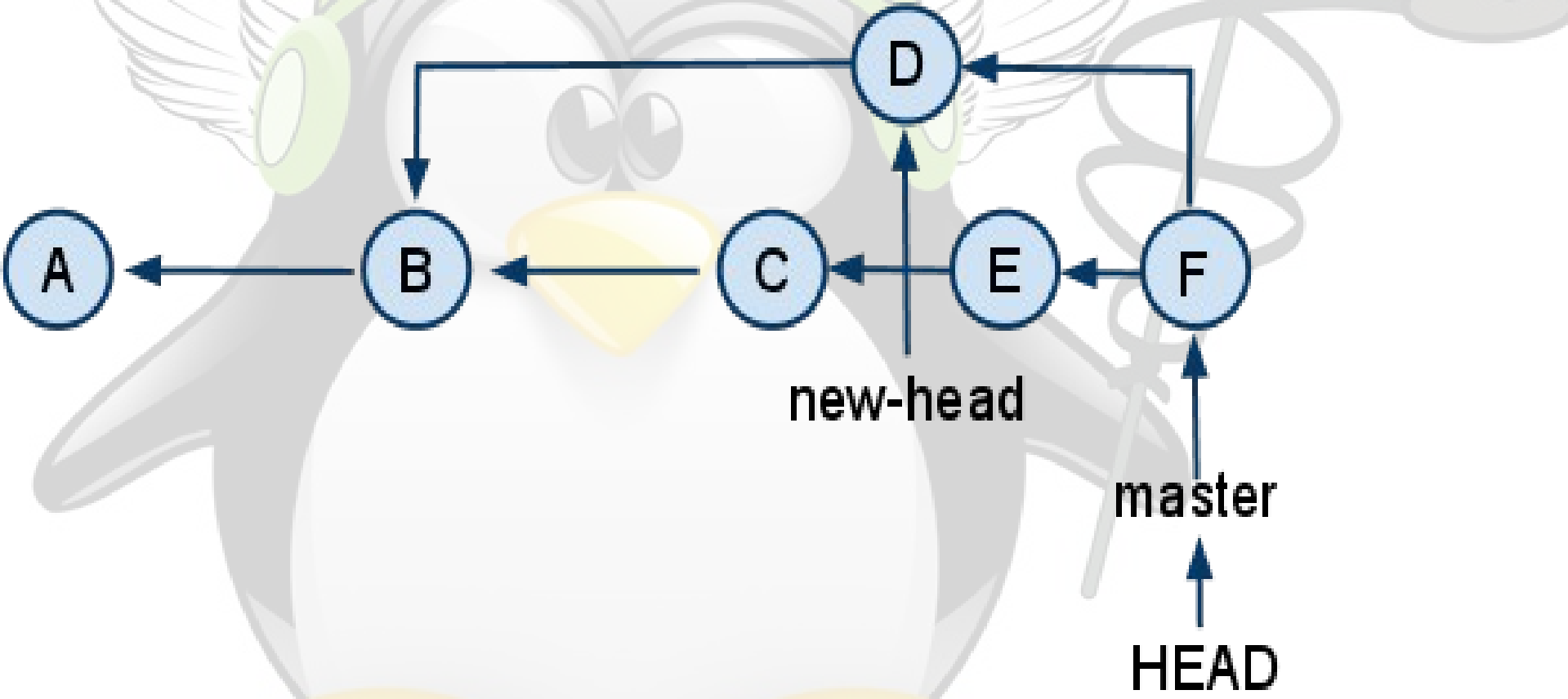


Διαγραφή κλαδιών

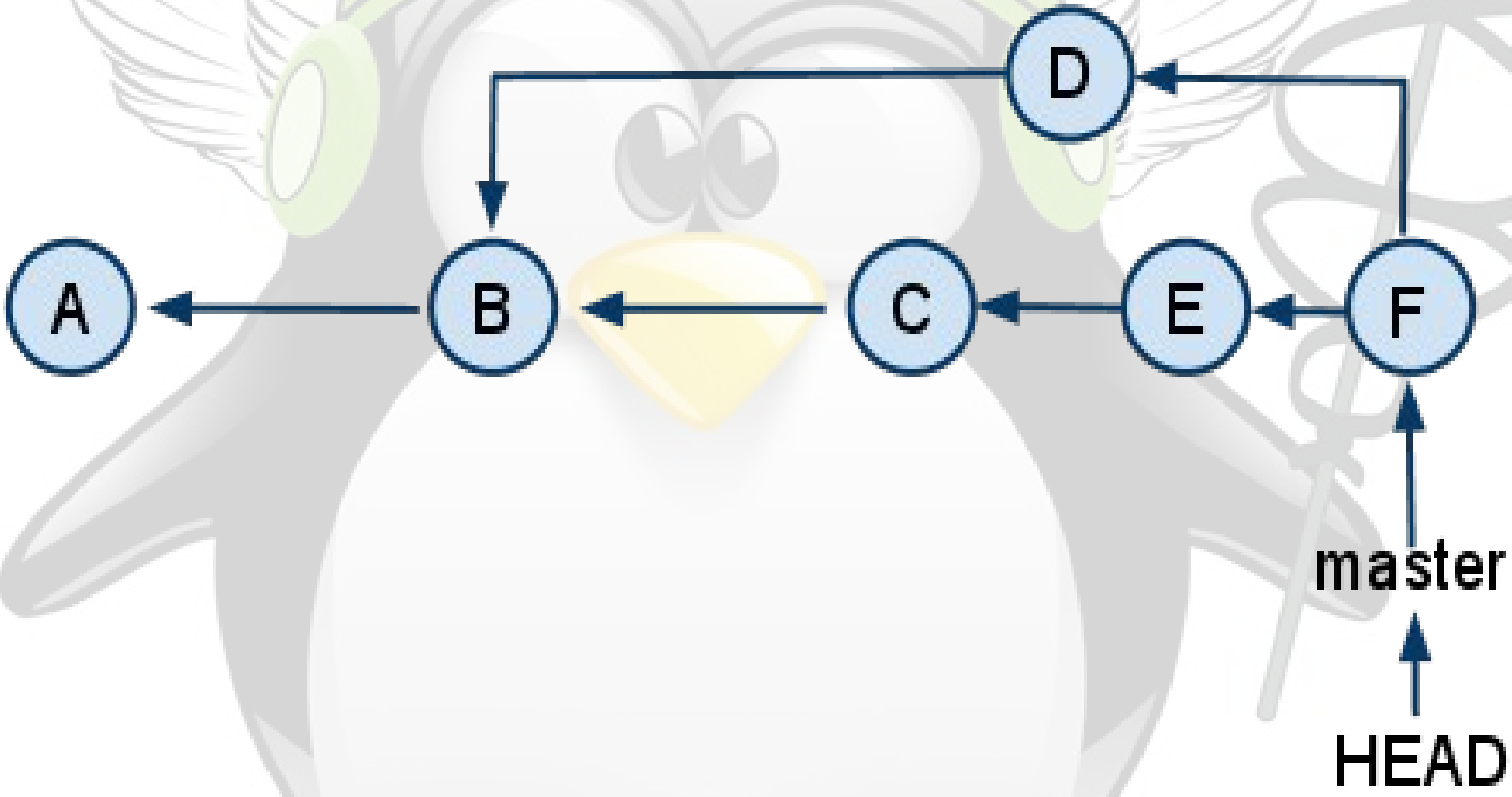
Διαγραφή κάποιου branch όταν πάψει να μας χρειάζεται – αφού το έχουμε κάνει merge με το master

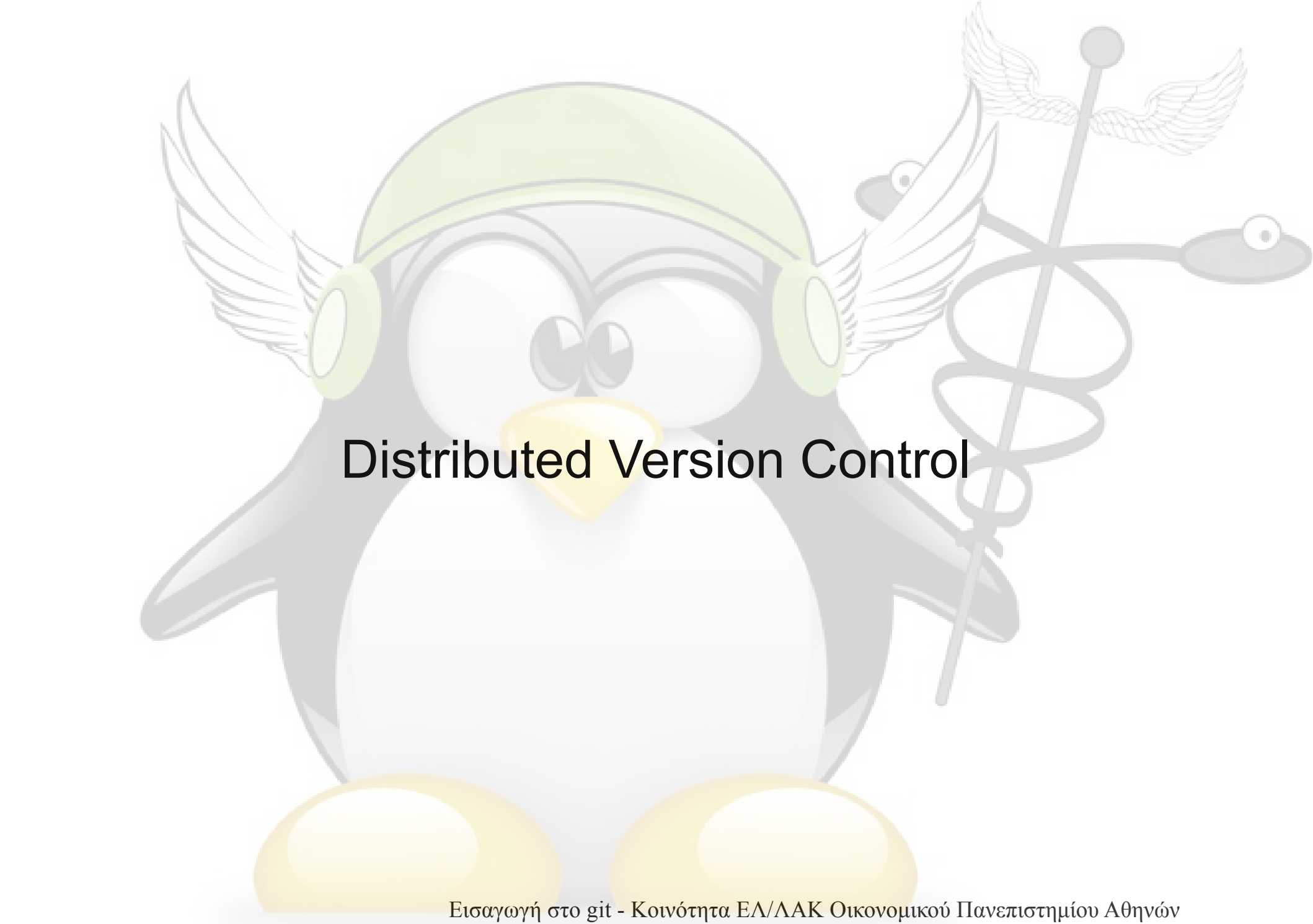
git branch -d [head]

Έστω



git branch -d new-head





Distributed Version Control

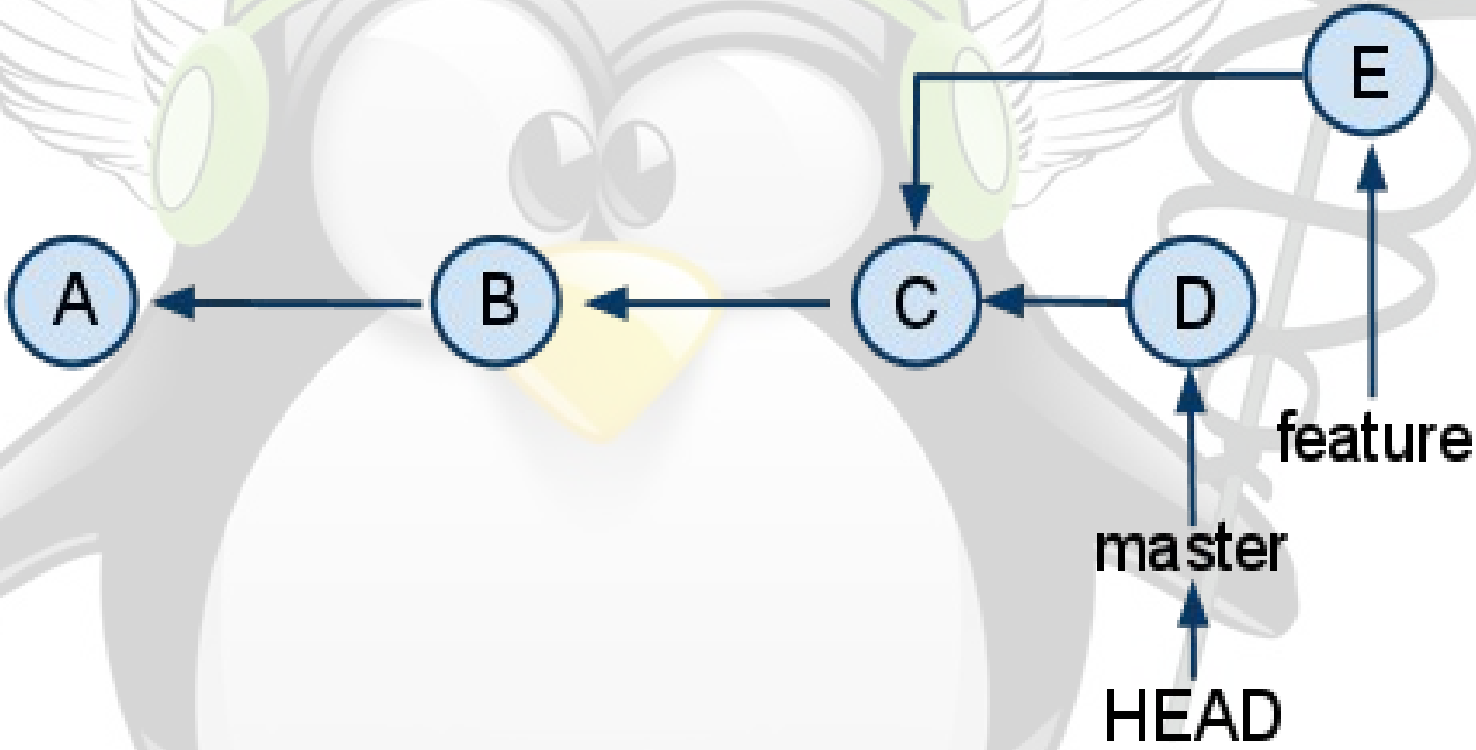
Αντιγραφή του repository

git clone [project-path/project-name]

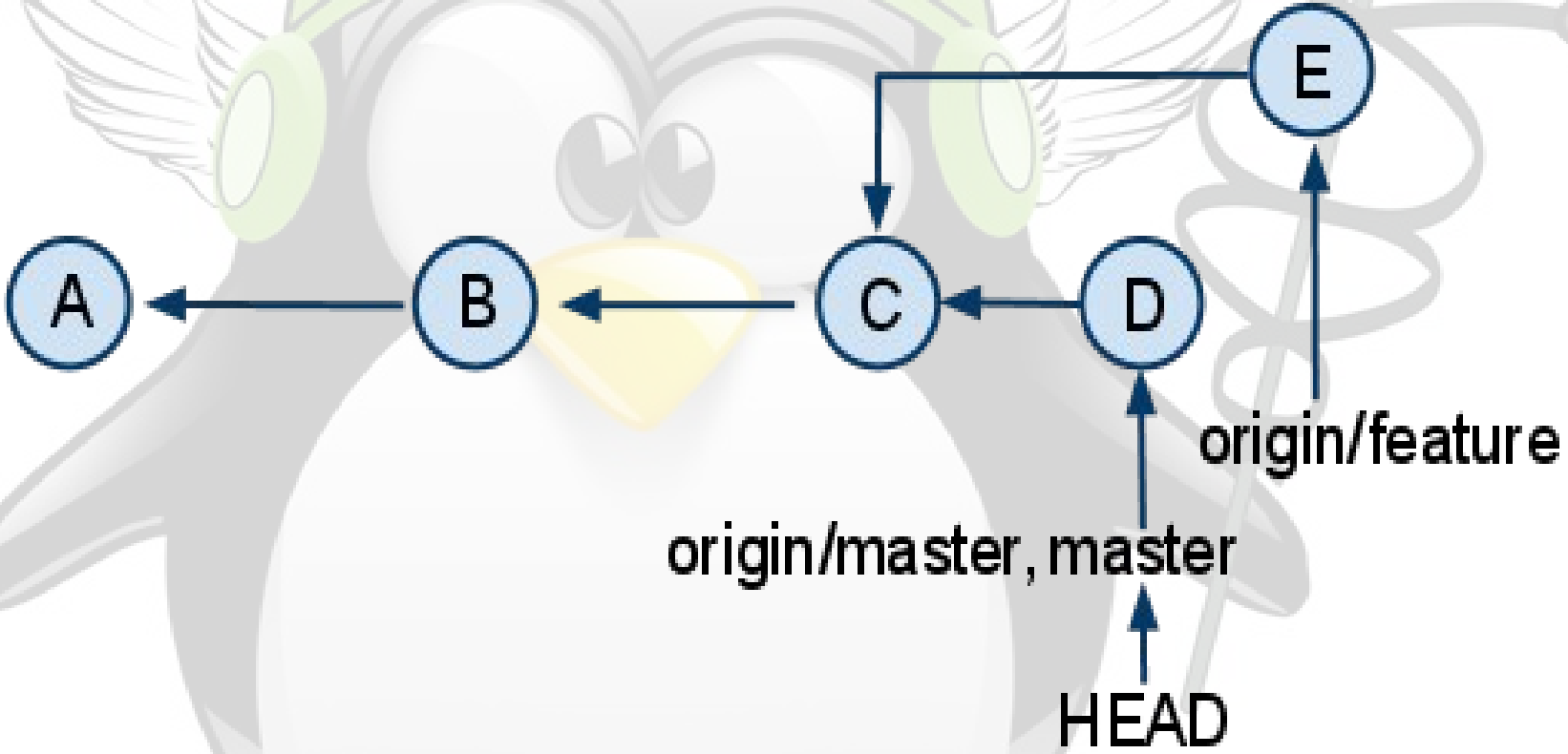
Αντιγραφή του repository

- Δημιουργία directory “project-name” και αρχικοποίηση του repository
- Αντιγραφή όλων των commits στο νέο repository
- Προσθήκη αναφοράς απομακρυσμένου repository με το όνομα origin
- Προσθηκη απομακρυσμένων heads με το όνομα original/[head-name]
- Δημιουργία head που να δείχνει στο origin/[curent-head-name], που είναι αυτό που ήταν ενεργό κατά την αντιγραφή του repository

Έστω το repository του συνεργάτη



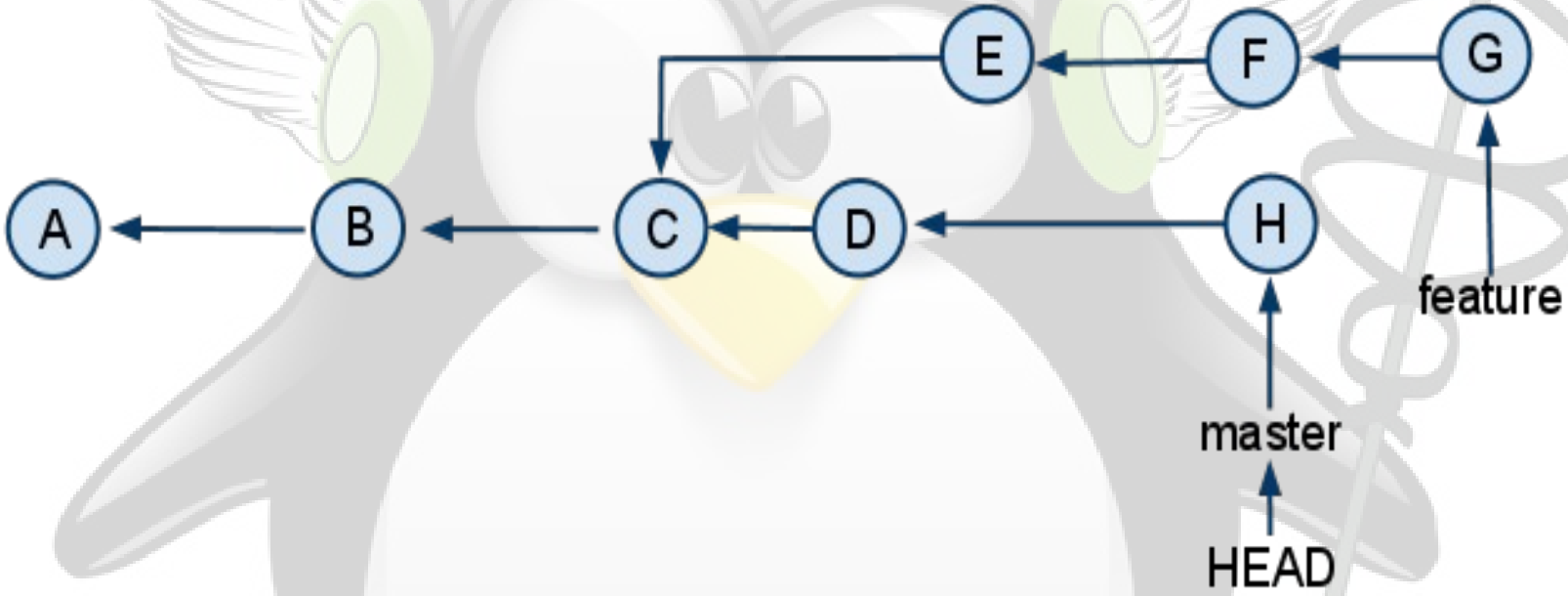
Μετά το cloning το δικό μας γίνεται



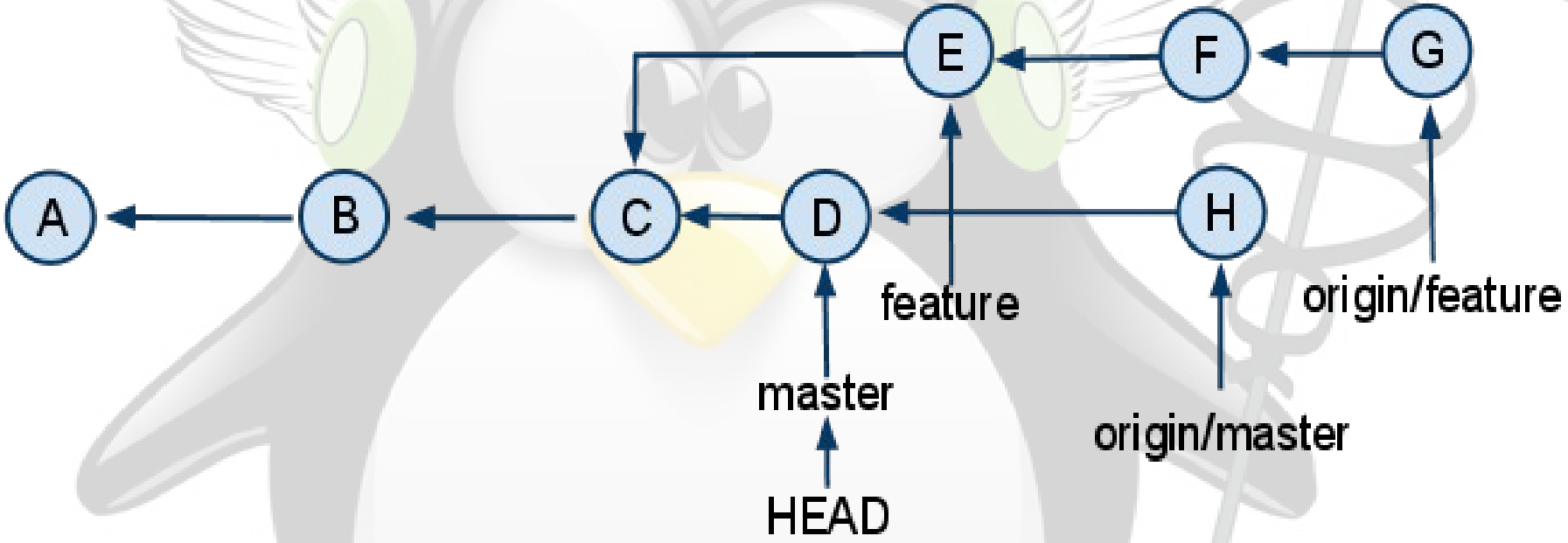
Λήψη αλλαγών

git fetch [remote-repository-reference]

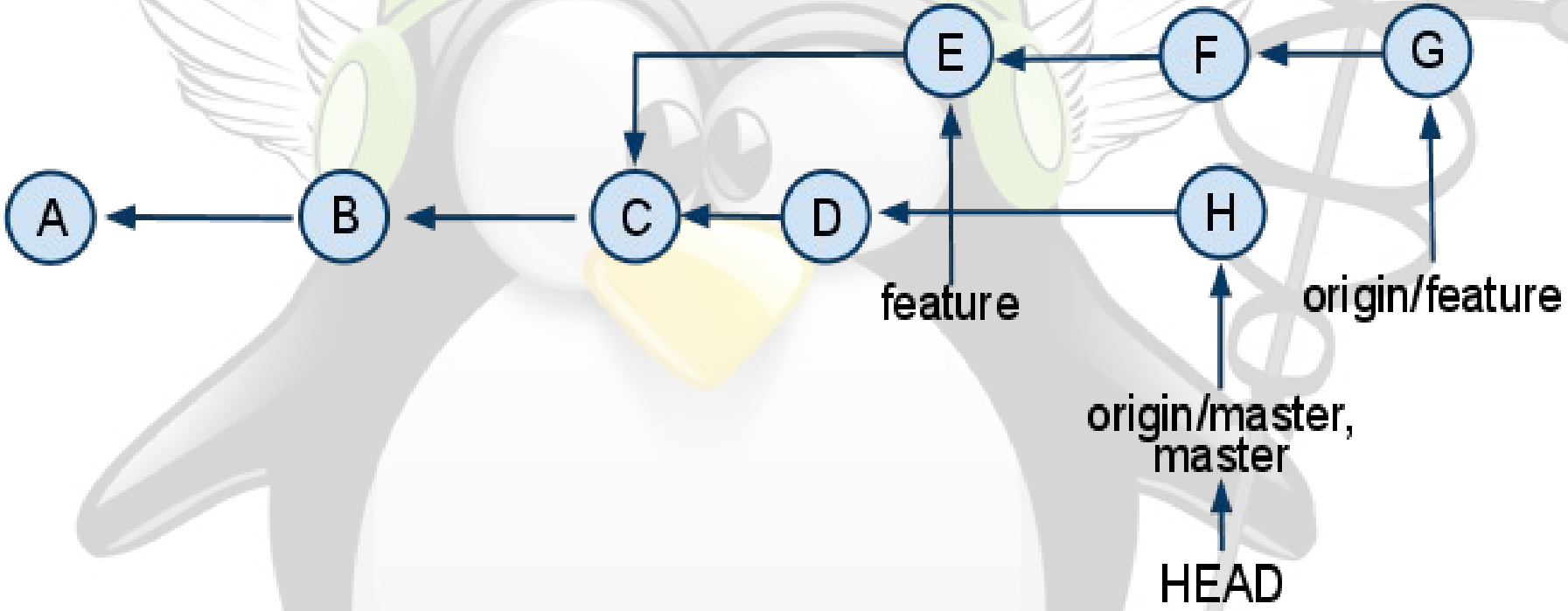
Έστω το repository του συνεργάτη μας



Και έχουμε


A
B
C
D
E
F
G
H
feature
origin/feature
master
origin/master
HEAD

Git pull [remote-repository-reference] [remote-head-name]



Αποστολή αλλαγών στο απομακρυσμένο repository

git push [remote-repository-reference] [remote-head-name]

Το Git κάνει το εξής στο απομακρυσμένο repository:

- Προσθέτει τα νέα commits του τοπικού repository
- Βάζει το [remote-head-name] να δείχνει στο ίδιο commit με το τοπικό repository

Πρόσθεση και Διαγραφή απομακρυσμένων κλαδιών

Πρόσθεση κλαδιού

1. ***git push origin new-branch***
   git checkout [some-other-branch]
   git branch -f new-branch origin/new-branch
   git checkout new-branch
2. ***git push --set-upstream origin new-branch***

Διαγραφή κλαδιού

git push [remote-repository-reference] :[head-name]

Git με κεντρικό repository

Οι προγραμματιστές κάνουν απλά τη δουλειά τους κάνοντας push/pull στο κεντρικό repository.

Git hosting

http://repo.or.cz

Περισσότερο διάβασμα!

- http://www.eecs.harvard.edu/~cduan/technical/git/
- http://git-scm.com/
- http://help.github.com/git-cheat-sheets/
- http://www.git.or.cz/course/svn.html
- Git Ready
- Pro Git
- Git Reference
- Git community Book
- ….

Foss Aueb

http://foss.aueb.gr/

irc: #foss-aueb @ freenode
[http://foss.aueb.gr/irc/]