



**git**



# Σεμινάριο Git & GitLab

Θεοδώρα Παναγέα

dpanagea@di.uoa.gr

# Περιεχόμενα

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

# Τι είναι το git

- Open-Source σύστημα ελέγχου εκδόσεων
- Εργαλείο από command line
- Πολλά Git-repository managers, πχ GitLab, GitHub κλπ



# Πλεονεκτήματα Χρήσης Git

- Αποθήκευση πολλών εκδόσεων ενός αρχείου
- Μετάβαση σε προηγούμενη, πιο stable έκδοση
- Συνεργασία με πολλά άτομα στο ίδιο project
- Παράλληλη επεξεργασία από πολλούς χρήστες
- Πληροφορία για τις μεταβολές σε ένα αρχείο
- Backup για την εργασία μας



# Εγκατάσταση του git

- Debian, Ubuntu
  - # apt install git
- Fedora
  - # yum install git
- Mac
  - <http://git-scm.com/download/mac>
- Windows
  - <https://git-scm.com/download/win>

# Βιβλιογραφία

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

# Ρύθμιση του Git

- `git config --global`
  - Αλλάζει τις ρυθμίσεις του git

```
$ git config --global user.name "Name Surname"
```

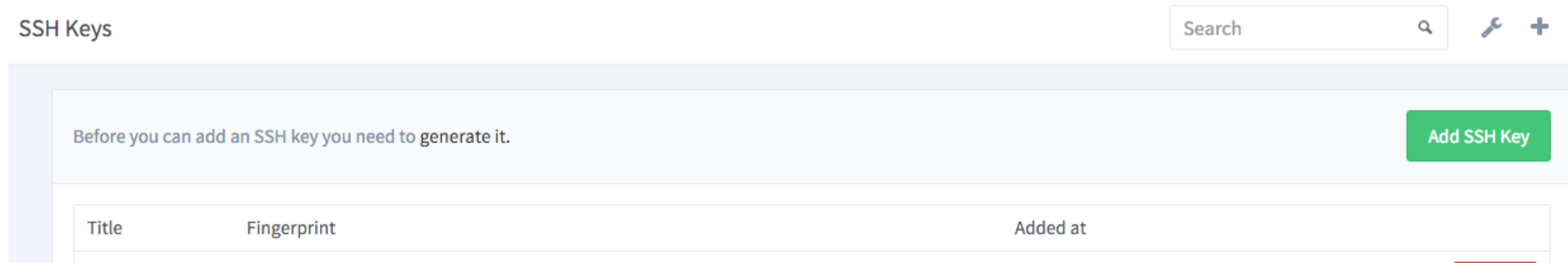
```
$ git config --global user.email "user@di.uoa.gr"
```

- Δημιουργία των ssh-keys
  - Ένας τρόπος για να συνδέεσαι στο git χωρίς να ζητάει συνέχεια κωδικό




```
$ ssh-keygen -t rsa -b 4096 -C 'user@di.uoa.gr'
```

# Add new ssh -keys GitLab

- Προσθήκη κλειδιών στο GitLab



SSH Keys

Search   

Before you can add an SSH key you need to generate it. [Add SSH Key](#)

Title	Fingerprint	Added at
-------	-------------	----------

- Προσθέτουμε το `id_rsa.pub` ώστε να μπορούμε να συνδεόμαστε μέσω ssh στο git



# Git Repositories

- Το μέρος όπου υπάρχει το project στο Git
- Περιήγηση στο φάκελο του πηγαίου κώδικα του project
- Δημιουργία ενός git repository με τις παρακάτω εντολές

```
$ git init
```

```
$ touch README.md
```

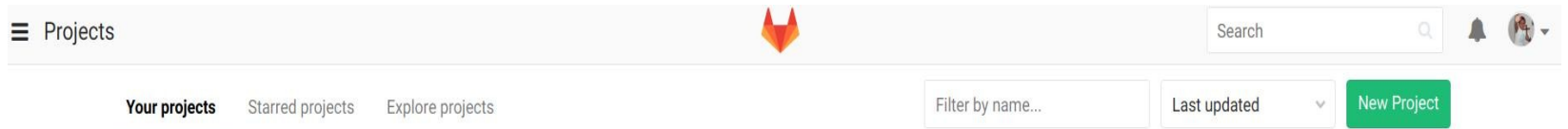
```
$ git add README.md
```

```
$ git commit -m "add readme"
```

```
$ git remote add origin git@git.scanlab.gr:~/name/git-project-name.git
```

```
$ git push -u origin master
```

# Μέσω γραφικού περιβάλλοντος





### New project

Create or Import your project from popular Git services

#### Project path

https://git.scanlab.gr/ georget

#### Project name

awesome\_project

Want to house several dependent projects under the same namespace? [Create a group](#)

#### Import project from

- GitHub
- Bitbucket
- GitLab.com
- Google Code
- Fogbugz
- Gitea
- git Repo by URL
- GitLab export

#### Project description (optional)

Description format

#### Visibility Level (?)

Visibility Level

- Private  
Project access must be granted explicitly to each user.
- Internal  
The project can be cloned by any logged in user.
- Public  
The project can be cloned without any authentication.

Some visibility level settings have been restricted by the administrator.

Create project

Cancel



Project 'awesome\_project' was successfully created.



# awesome\_project

☆ Star 0 SSH git@git.scanlab.gr:georget/awe + Global

- Members
- Deploy Keys
- Integrations
- Protected Branches
- Runners
- Variables
- Triggers
- CI/CD Pipelines
- Edit Project

## The repository for this project is empty

If you already have files you can push them using command line instructions below.

Otherwise you can start with adding a [README](#), a [LICENSE](#), or a [.gitignore](#) to this project.

You will need to be owner or have the master permission level for the initial push, as the master branch is automatically protected.

### Command line instructions



### Members

Add a new member to awesome\_project

Search for members by name, username, or email, or invite new ones using their email address.

[Read more about role permissions](#)

On this date, the member(s) will automatically lose access to this project.

#### Existing members and groups

Members with access to awesome\_project 1



**George Tsiatsios** @georget It's you  
Joined 15 minutes ago

Master



Users were successfully added.

### Members

Add a new member to awesome\_project

Search for members by name, username, or email, or invite new ones using their email address.

[Read more](#) about role permissions

On this date, the member(s) will automatically lose access to this project.

#### Existing members and groups

Members with access to awesome\_project 2 Find existing members by name Name, ascending

	<b>Dimitris Kassos</b> @dkassos Joined about a minute ago	Master	Expiration date	
	<b>George Tsiatsios</b> @georget <span>It's you</span> Joined 16 minutes ago			Master

# git status

- Δείχνει την κατάσταση του git
- Εκτελείς την εντολή για να ξέρεις την κατάσταση του τοπικού σου repo

\$ git status

*On branch master*

*Your branch is up-to-date with 'origin/master'.*

*nothing to commit, working directory clean*

# git add <file>

- Ζητάει από το git να προσθέσει ένα νέο αρχείο στο τοπικό repository
- Πρέπει να ακολουθείται από 'commit' ή ένα γενικό μοτίβο
- Παίρνει ως παράμετρο το νέο αρχείο ή ένα γενικό μοτίβο
  - . όλα τα αρχεία
  - '\*.txt' όλα τα αρχεία .txt



# git commit

- Ένα στιγμιότυπο του git
  - 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

# git help command

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

# Βασικό git workflow

- `touch 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>`

# git log

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



# 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

# 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

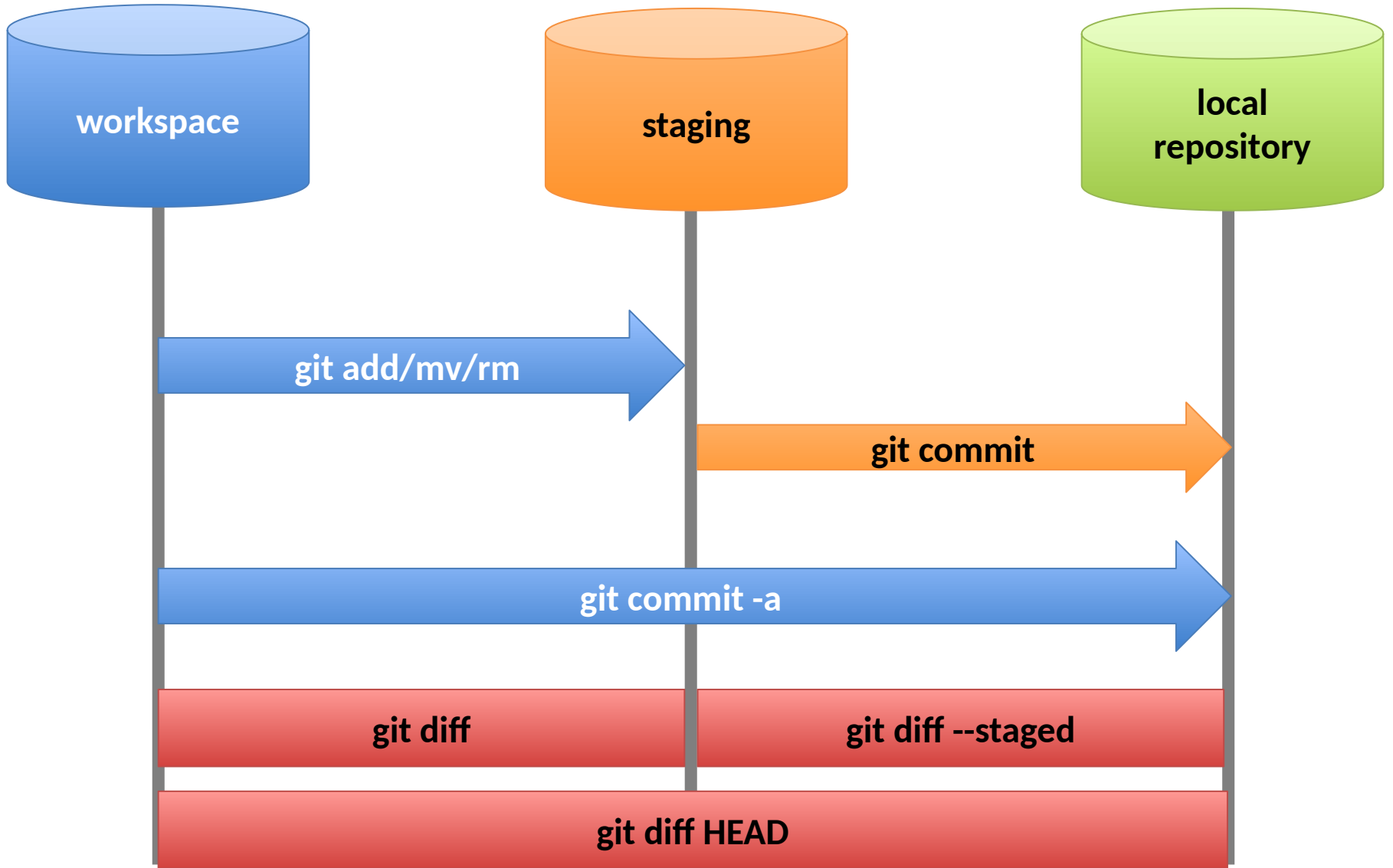
- Δείχνει τι αλλαγές έχουν γίνει που θα καταγραφούν στο επόμενο commit
  - Μας λέει τι έχει γίνει ήδη git add

# git diff HEAD

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

# Staging area

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



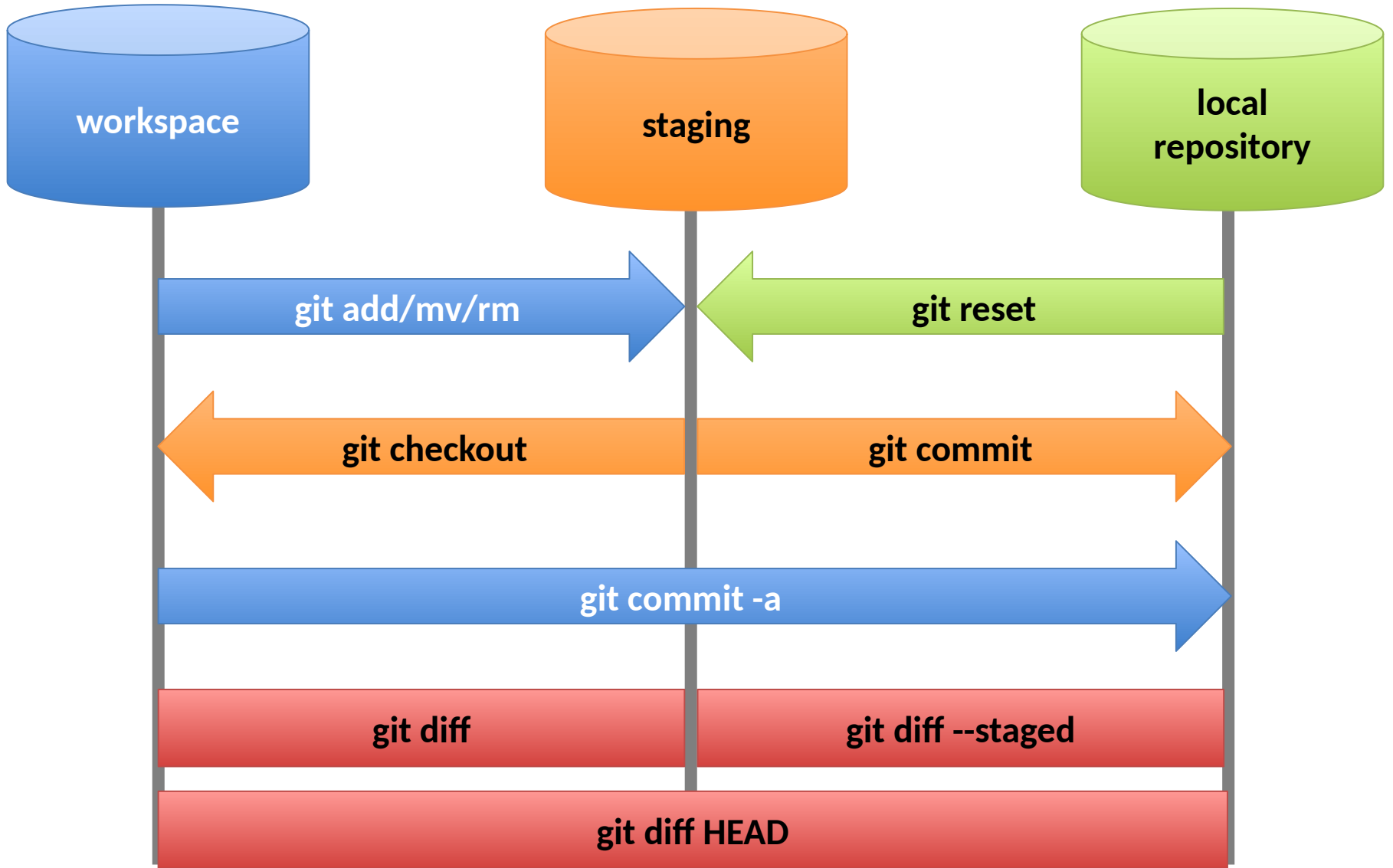
# git checkout <file>

- 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`



# 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

# git checkout -b <branch>

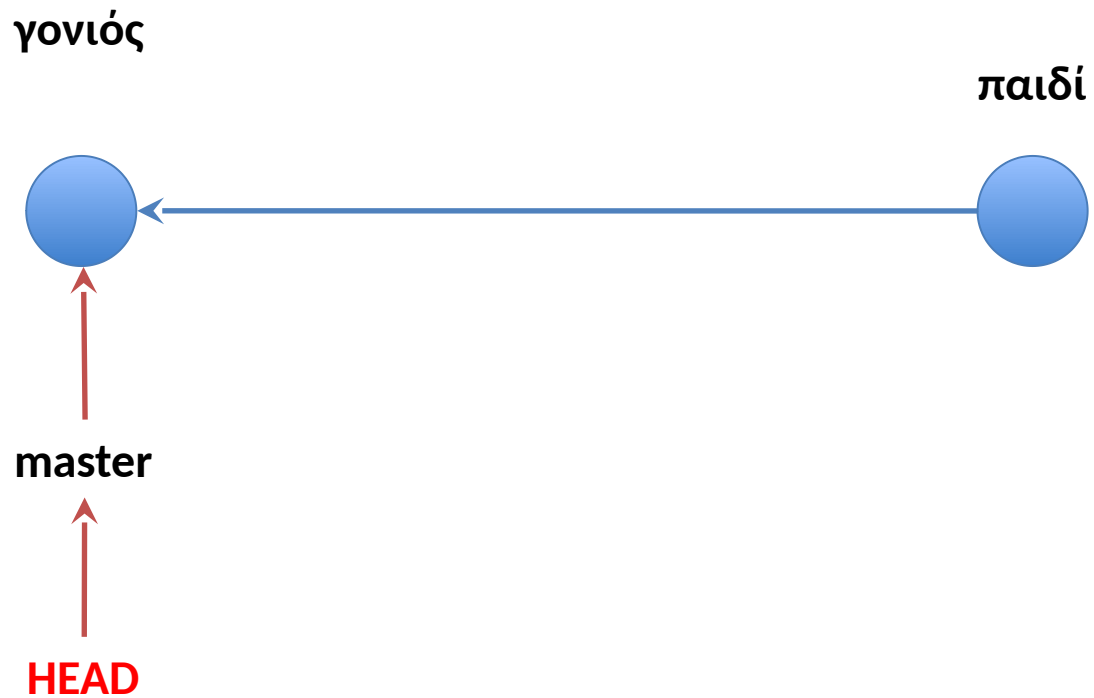
- Δημιουργεί νέο branch και αλλάζει το τρέχον branch σε <branch>
- git checkout -b <branch> σημαίνει:
  - git branch <branch>
  - git checkout <branch>



# 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

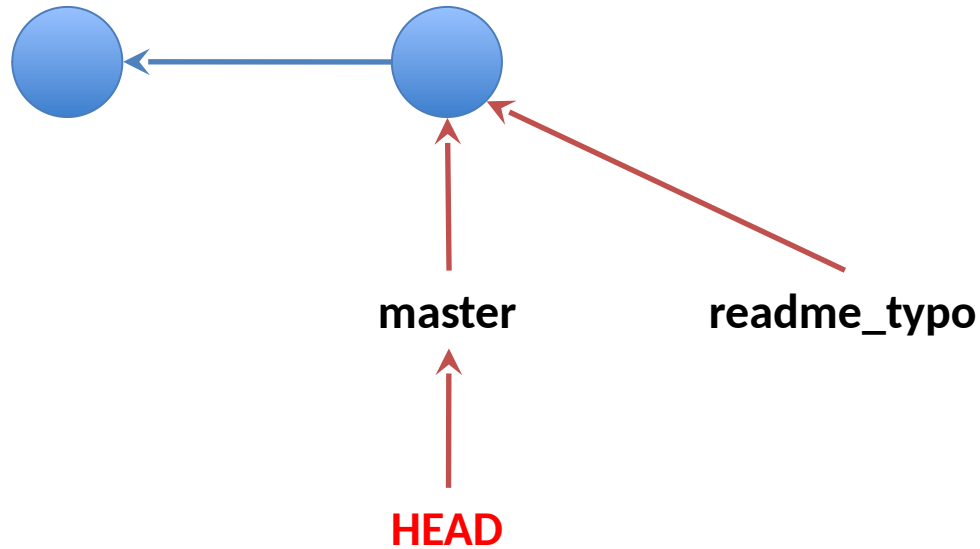
# git commit



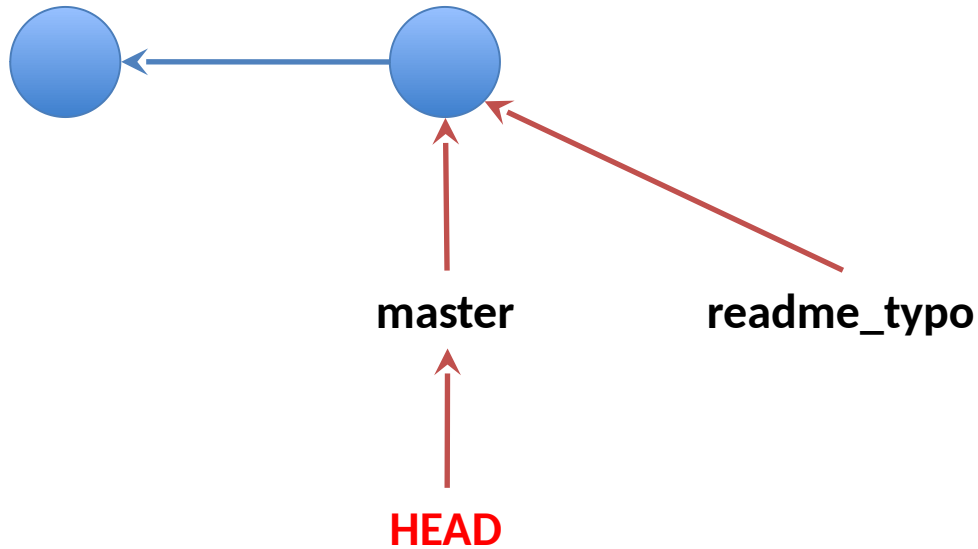
# git commit

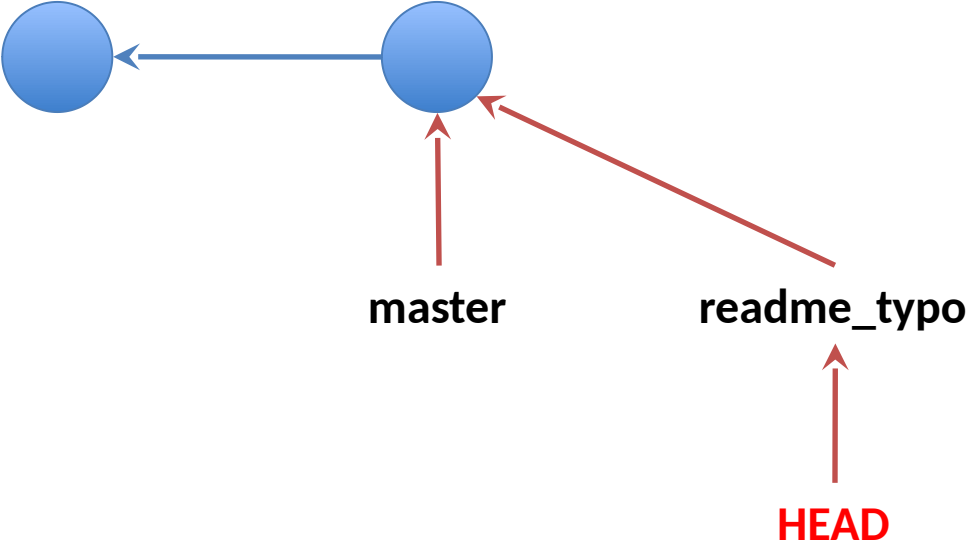
- Τώρα μπορούμε να μιλήσουμε για commits με όρους γράφων
- git commit
  - Δημιουργεί ένα νέο commit αντικείμενο
  - Ορίζει τον **γονιό** του να είναι το commit αντικείμενο στο οποίο δείχνει το τρέχον branch (δηλαδή το branch στο οποίο δείχνει HEAD)
  - Μεταφέρει το τρέχον branch στο νέο commit

# git branch readme\_typo

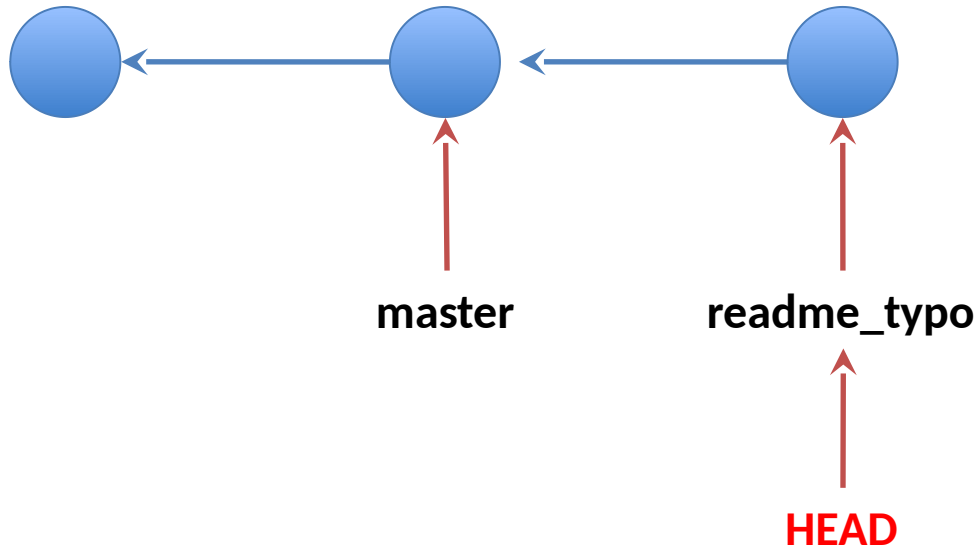


# git checkout readme\_typo





# 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



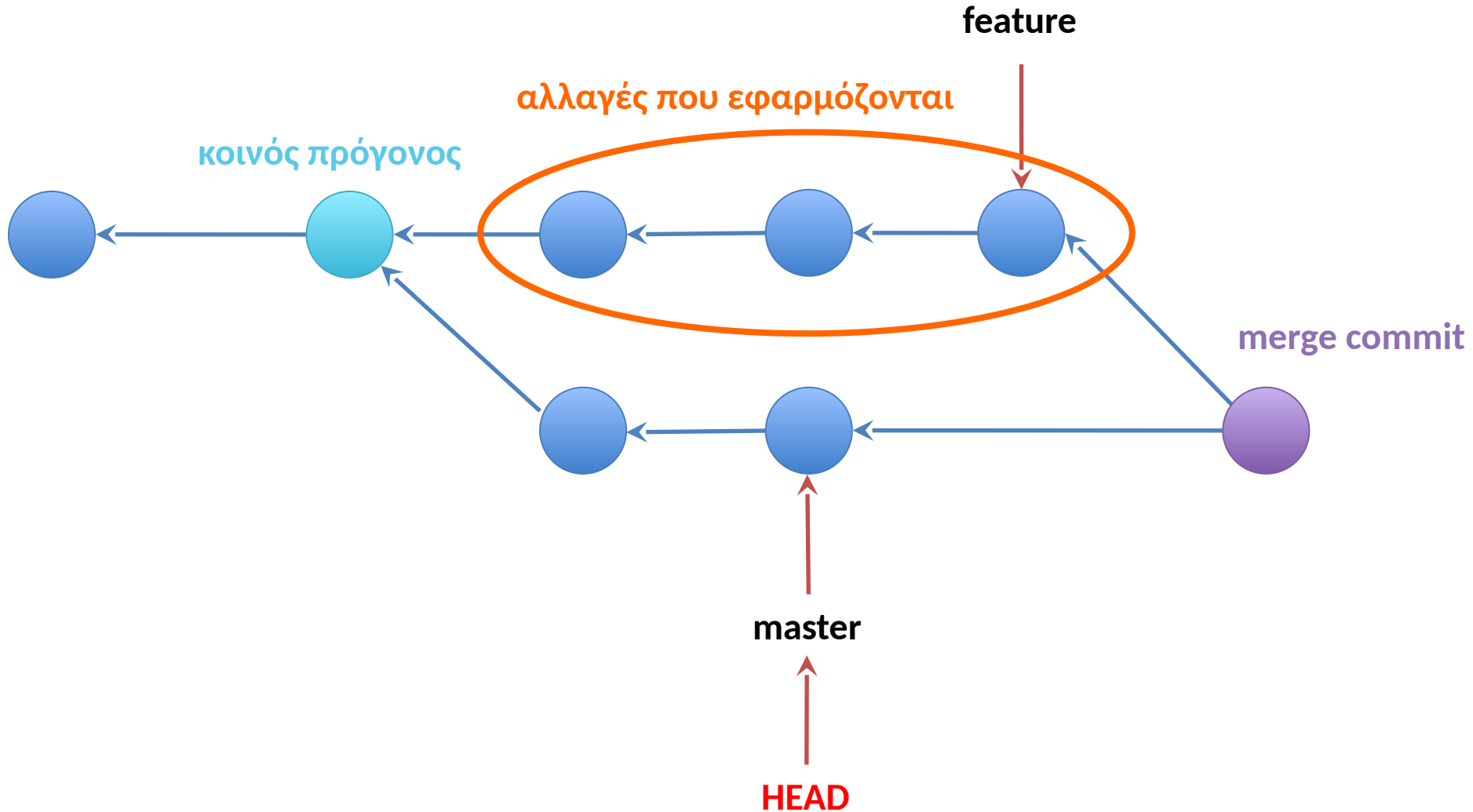
# Αλγόριθμος merging

- 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



# 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

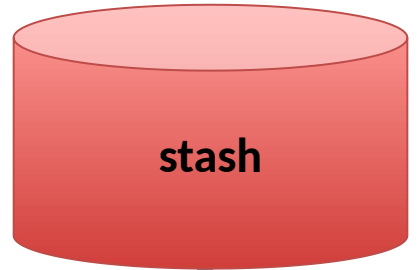
- Κρατάει στην άκρη τις αλλαγές:
  - στο working copy
  - στο staging area
- Καθαρίζει το working copy και το staging
- Πλέον μπορούμε να κάνουμε checkout ένα άλλο branch χωρίς να πρέπει να κάνουμε commit ή να χάσουμε τις αλλαγές μας

# git stash pop

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



workspace



stash



git stash



git stash pop



# GitLab

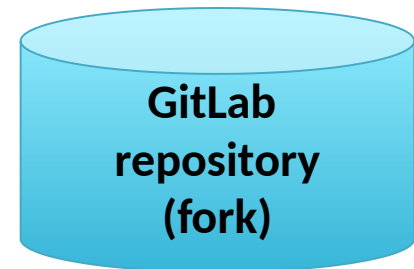
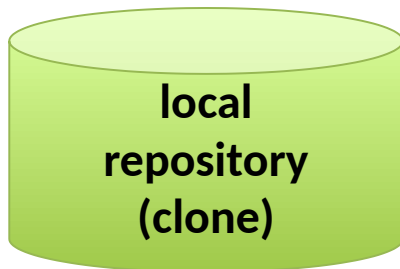
- Ένα website που μπορούμε να ανεβάσουμε αντίγραφα των repo μας και να συνεργαστούμε
- Προσφέρει εργαλεία για συνεργασία

# Ας συνεργαστούμε!

- Μπείτε στο <http://anargit.scanlab.gr/>
- Πατήστε “create project”
- 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
- Κάθε remote έχει:
  - το δικό του URL
  - ένα όνομα

# 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`





# 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 εν τω μεταξύ
- Για να μη 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!!