Git

“I find your lack of faith disturbing”
Darth Vader

Jo Colina

@jsmrcaga

GitHub

https://education.github.com/pack

https://github.com/jsmrcaga/workshop-utc

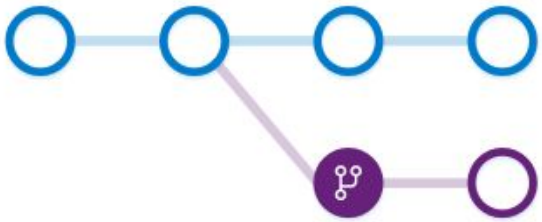
10 M

Projects on Github

2013 - github.com

Git

Version Control System (VCS)



- Gestion de versions
- Gestion de conflits
- Gestion d’historique
- Gestion de déploiement

- Code Maintenable
- Code Collaboratif
- Projets Propres
- Projets Compréhensibles

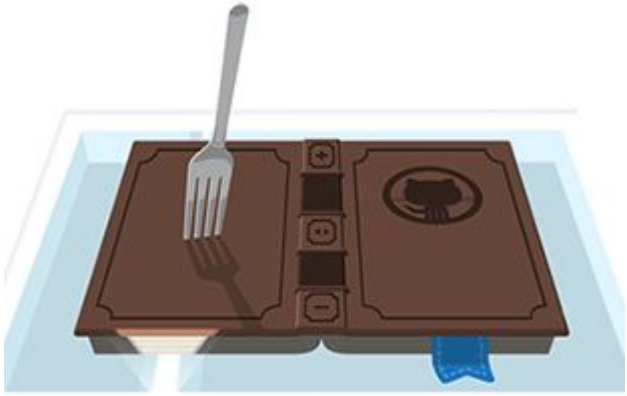
Notions

... de base

- Dépôt
- Clonage
- Fork
- Timeline

Dépôt + Clonage - Fork

- Dépôt = projet versionné
  - ⇒ Historique
  - ⇒ Visibilité (blaming)
- Clonage
  - ⇒ Copier un dépôt
- Fork
  - ⇒ Créer un embranchement

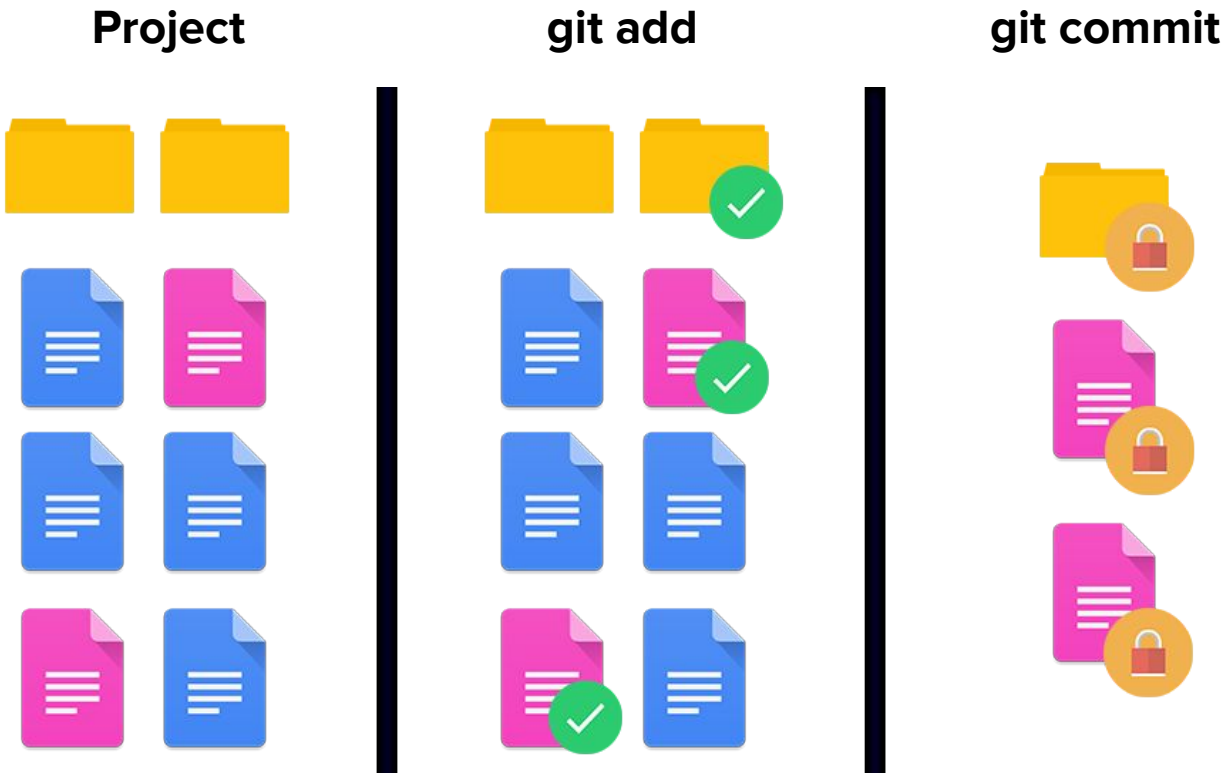


®GitHub

Timeline

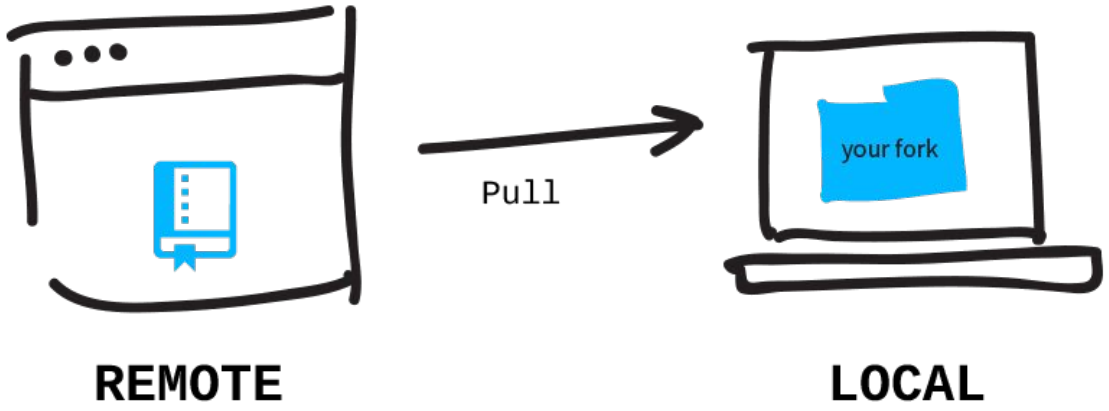
Git Workflow 101

Add / Commit



Pushing / Pulling

- **REMOTE** ⇒ Dépôt dans le cloud (~= lien)
- **PUSH** ⇒ Envoi de code dans le REMOTE
- **PULL** ⇒ Demande de code depuis le REMOTE



Syntaxe simple

```
git add . | -A | -u | <file list>
git commit -m “<message>” [-m “<desc>”]

git pull [remote] [branch]
git push [remote] [branch]

git reset
```

“Aye ho! Aye ho! It's off to work I go!”

7 Dwarves

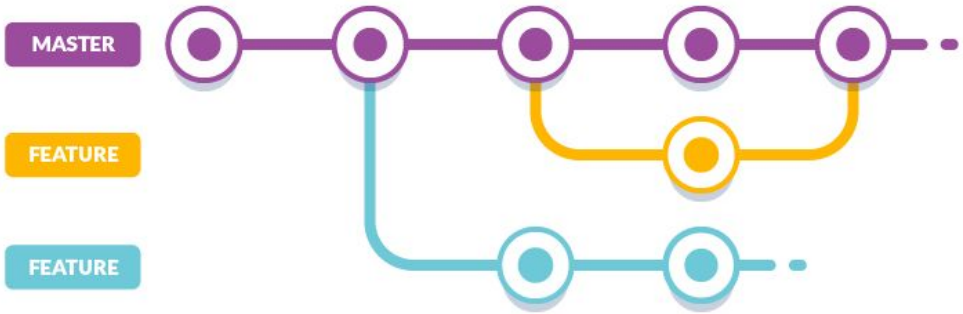
Branches

Déjà?

- Nouvelles fonctionnalités
- Imperméabilité
- Historique
- Lisibilité

http://learngitbranching.js.org/

Branches



- **Branche** ⇒ Nouvelle feature
- **Commit** ⇒ Noeud dans l'historique de la branche
- **Merge** ⇒ Mélange de deux branches

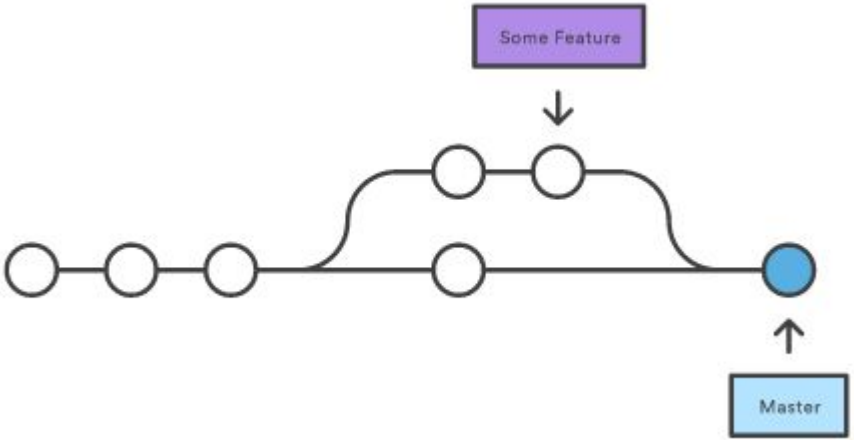
Syntaxe simple

```
git branch [name]
git checkout <name>

git checkout -b <name>

git push --set-upstream | -u <remote> <branch>
```

Merging

Merging ⇒ La rencontre des deux branches | automatique!!

```
git pull [--rebase] <branch name>
```



“If people knew how hard I worked to get my mastery, it wouldn’t seem so wonderful after all”

Michael Angelo

Travail en équipe

Maiiis! T'as tout cassé!

- Modification simultanée
- Changements importants
- Différents besoins

What? - Conflicts

```
87
88      'gh-pages': {
89  <<<<<<< HEAD
90        options: {
91          base: 'site/build',
92          repo: 'git@github.com:bocoup/moebio_framework.git'
93        },
94        src: '**/*'
95      }
96  =======
97         options: {
98           base: 'site/build',
99           repo: 'git@github.com:bocoup/moebio_framework.git'
100        },
101        src: '**/*'
102     },
103
```

Conflicts

```
<<<<<<< HEAD
    // blablablabla
=======
    // different blablabla
>>>>>>> Dummy commit
```

Il suffit de choisir le commit le le plus jeune ou le plus adapté et nécessaire.

2 Solutions Simples

`git stash / apply`

`stash :` Élimine les changements **non-staged** (qui n'ont pas été ***add*** ni ***commit***) et les garde dans un “coffre”.

`apply :` Reprend les dernières modifications et les réapplique pour pouvoir les ***stage/push***

`git pull --rebase`

Idem que ***`git stash / git apply`*** mais avec un commit et tout seul!

“I have not failed. I've just found 10,000 ways that won't work.”

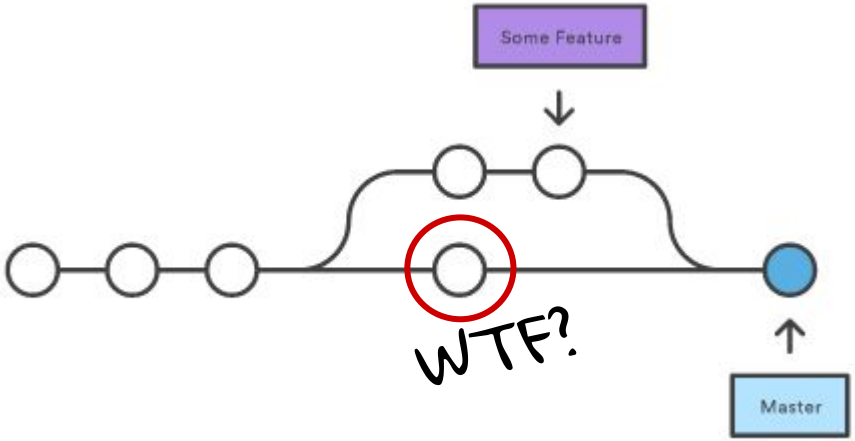
Thomas Edison

Rebasing

The way I see it, if you're gonna build a time machine into a car, why not do it with some *style?*

- Modification historique
- Empêcher les conflits
- Faciliter les merge

git rebase



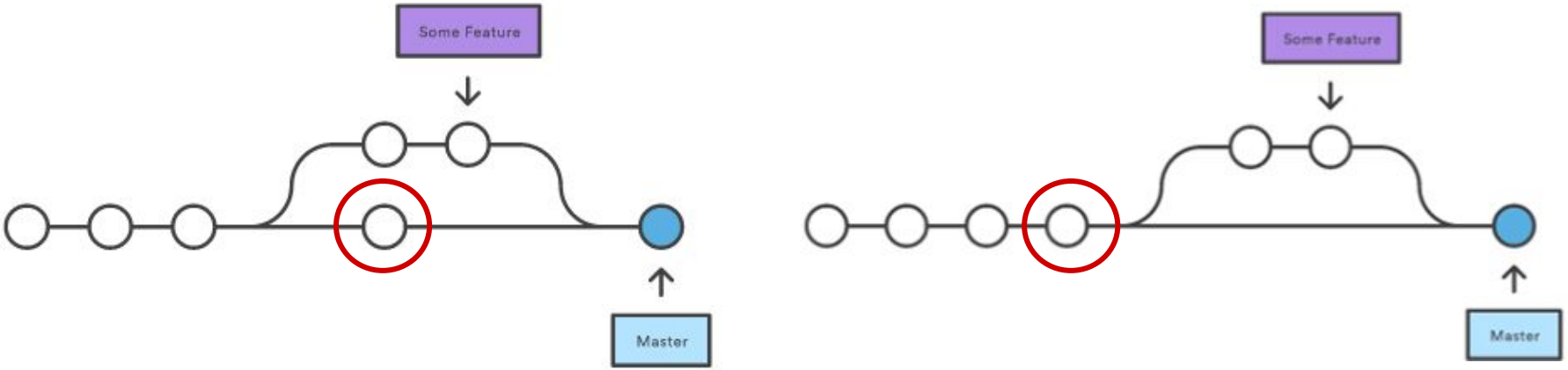
⇒ Merge

⇒ Merge Commit

⇒ Pas ouf...

git rebase

```
git [pull] [--]rebase <from_branch>
```



Rebase ⇒ Données à jour!

`git fetch`

“What happens to us in the future? Do we become assholes or something?”

Marty McFly

Oups...

Mon tuteur va me tuer...

- Réparation commits
- push -f
- Cherry Pick

git reset

- Permet d'enlever des fichiers du staging (`git add`)
- Permet de réinitialiser une branche
- Permet de revenir sur un commit

Syntaxe simple

```
git reset

git reset --hard <[origin/]branch>|<commit>

git reset HEAD^|[~nb]
```

git push -f

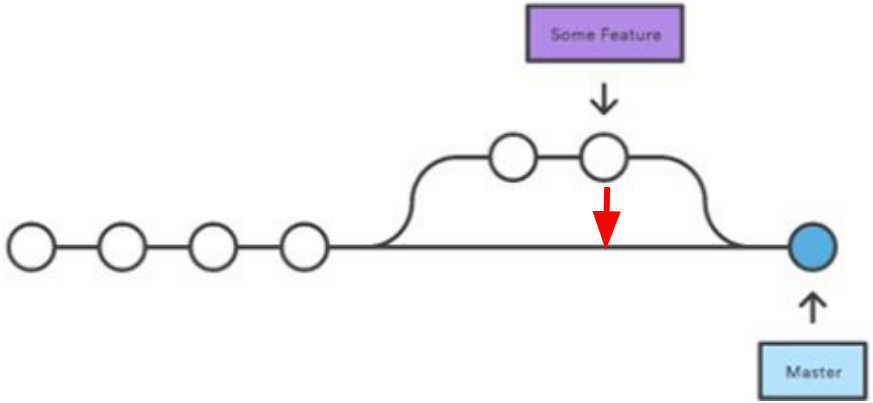
`-f` pour *force*

/!\ Ne pas utiliser sans tester!

- Permet de **`push`** dans le **`remote`** sans vérifier l'intégrité des données. S'il existe des différences, elles **seront ignorées**.

git cherry-pick <commit>

- Permet d'intégrer le commit donné dans la branche courante



“You shall not pass!”

Gandalf

Squash

What sorcery is this?

- Lisibilité de l'historique
- Lisibilité du code
- Facilite les merge

git rebase -i

Permet de *rebase* la branche courante sur la branche courante. (what?)

```
pick 838008d create resetCalendar() function to erase saved calendar data
pick 6cf60c8 fix typo in resetCalendar return statement
pick 6beff16 build out markup for Edit Calendar form

# Rebase 00c328b..6beff16 onto 00c328b
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#  x, exec = run command (the rest of the line) using shell
#
```

Syntaxe simple

```
git log
```

```
git rebase -i HEAD~<nb commits>
```

Go further

OMG

- git reflog
- git push -u
- git --amend
- git blame
- git tag
- Hooks

git reflog

⇒ Log de toutes les références de git.

- Permet de revenir sur une modification spécifique

```
33ee538 HEAD@{0}: rebase finished: returning to refs/heads/my_topic_branch
33ee538 HEAD@{1}: rebase: Making the newest commit in my_topic_branch.
76af5ab HEAD@{2}: checkout: moving from my_topic_branch to 76af5ab8a1839ec4f3e8a0e4b07507e7e94791d7^0
244eaad HEAD@{3}: commit: Making the newest commit in my_topic_branch.
960e5b3 HEAD@{4}: checkout: moving from dev to my_topic_branch
76af5ab HEAD@{5}: commit: Making a newer commit in dev.
960e5b3 HEAD@{6}: checkout: moving from my_topic_branch to dev
960e5b3 HEAD@{7}: checkout: moving from master to my_topic_branch
960e5b3 HEAD@{8}: checkout: moving from dev to master
960e5b3 HEAD@{9}: checkout: moving from master to dev
960e5b3 HEAD@{10}: commit (initial): Initial commit in master.
```

git push -u

`git push -u <remote> <branch>`

⇒ Dit a Git quelle branche utiliser dans le remote.

⇒ À utiliser lorsque le tout premier `commit` est fait manuellement

git --amend

⇒ Permet de changer le dernier commit

Hooks

⇒ Effectuer des actions en dépendant du dépôt

- Dès qu'il y a un push
- Dès qu'il y a un merge

Exemple:

- Tâches de tests automatiques (Jenkins, Travis)
- Déploiement

git blame

Public Shaming 101

⇒ Permet de voir qui a commit quelle ligne.

git tag (-a v1.4)

FINALEMENT!

⇒ Permet de donner une version au code actuel

54.12.22

Major Value	Minor Value	Build Value

Merci!