Git

(survival kit)

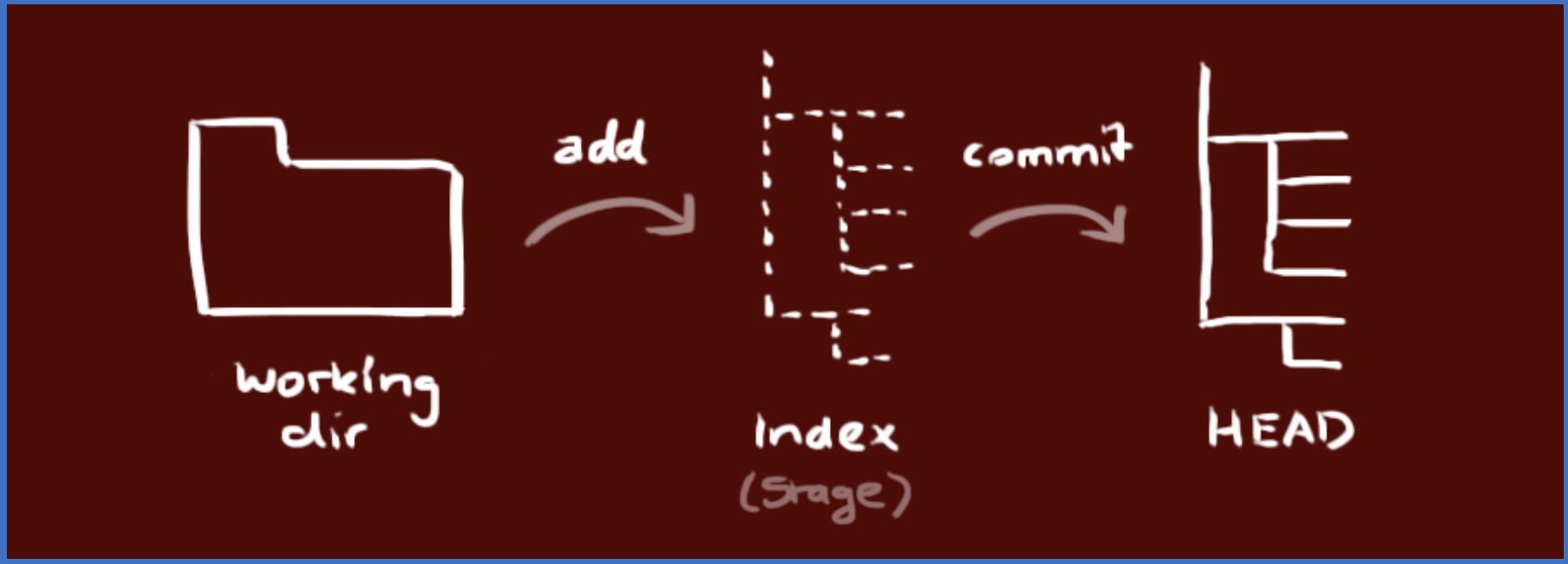
Workshop

Antoine Froger

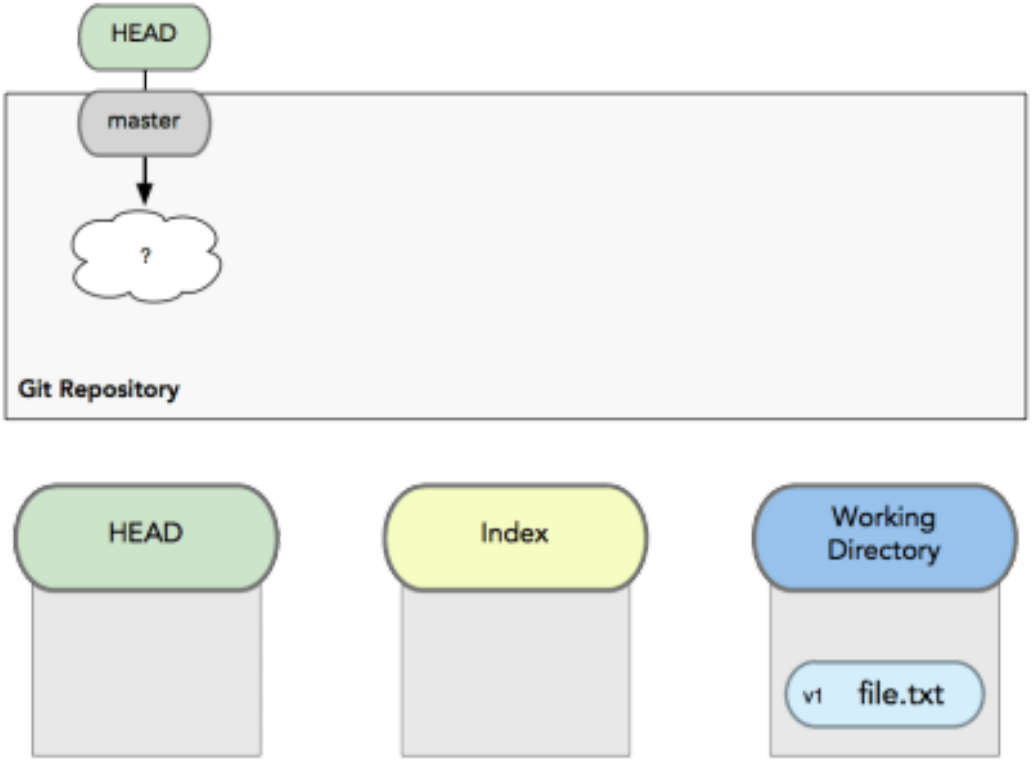
Workflow

The 3 trees

`The Working directory`	Sandbox
`The Index (or stage)`	Proposed next commit snapshot
`The HEAD`	Last commit snapshot

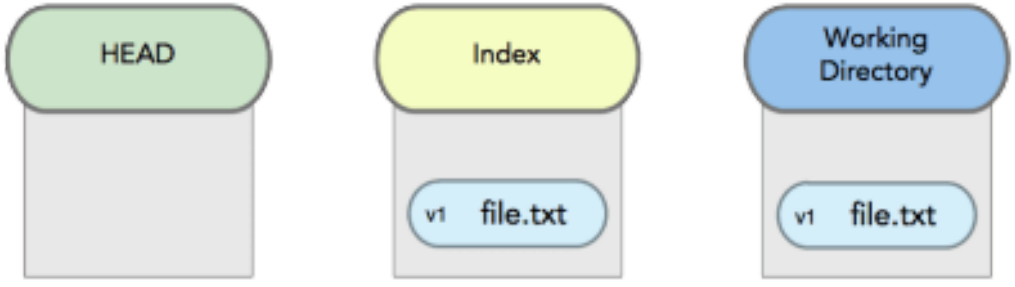


New file


HEAD
master
?
Git Repository
HEAD
Index
Working Directory
v1
file.txt

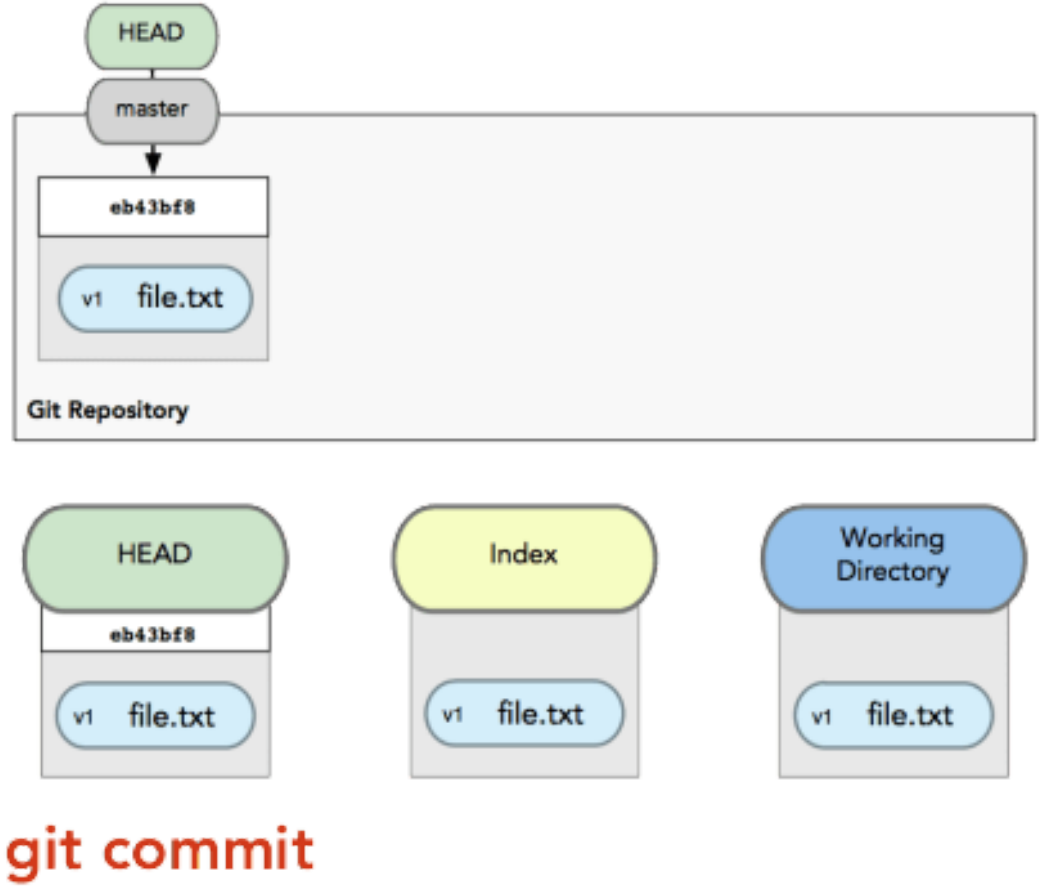
Add


HEAD
master
?
Git Repository


HEAD
Index
Working Directory
v1 file.txt
v1 file.txt

git add

Commit



git commit

Clean branches

Display branches

Local branches

```
$ git branch
* master
  rebase-i-demo
  untracked-demo
```

All branches (local and distant)

```
$ git branch -a
* master
  rebase-i-demo
  untracked-demo
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
```

Oups...

Wrong name ?

```
// Rename branch
$ git branch -m branch-old-name branch-new-name


// Rename branch
$ git branch -m branch-new-name
$ git branch -a
  master
  rebase-i-demo
  untracked-demo
* branch-new-name
```

Delete branch

Don't need this branch anymore

```
$ git branch -d obsolete-branch
// equivalent to git branch --delete
error: The branch 'obsolete-branch' is not fully merged.
If you are sure you want to delete it, run 'git branch -D
obsolete-branch'

$ git branch -D obsolete-branch
// Shortcut for --delete --force
Deleted branch obsolete-branch (was 12345)
```

Usefull options

-vv

Displays some information about the last commit

```
$ git branch -vv
* master        a3d3414 [origin/master] Add homepage
  rebase-i-demo a15dc16 Revert "Create documentation"
...
```

--no-merged

Only displays unmerged branches

```
$ git branch --no-merged
  rebase-i-demo
```

--merged

Only displays merged branches

```
$ git branch --merged
* master
  untracked-demo
```

Atomic commits

one commit
=
one responsability

easier to understand

easier to review

easier to revert

Detailed status

Show untracked files

```
$ git status
...
Untracked files:
...
    README.md
    vendor/
$ git status -u
$ // git config --global status.showUntrackedFiles all
$ // git status
...
Untracked files:
...
    README.md
    vendor/autoload.php
    vendor/composer/ClassLoader.php
    vendor/composer/LICENSE
```

Show your changes

Unstaged changes

```
$ git diff
```

Staged changes

```
$ git diff --staged
```

Without whitespaces

```
$ git diff -w
```

Add files to the stage

A specific file

```
$ git add README.md
$ git status
...
Changes to be committed:
...
    new file: README.md
```

All files

```
$ git add .
$ // git add --all
$ git status
...
Changes to be committed:
    new file: README.md
    new file: composer.json
...
```

Add files to the stage

Only add a fragment

Because you want atomic commits and not catch-all commits

```
$ git add -p README.md
...
# Git (for beginners)
+ This repository is dedicated to the git workshop
...
Stage this hunk [y,n,q,a,d,/,s,e,?]? y
...
## Commits
## Pushes
+## Branches
...
Stage this hunk [y,n,q,a,d,/,s,e,?]? y
```

Unstage files

One file

```
$ git reset README.md
Unstaged changes after reset
M    README.md
```

All the files

```
$ git reset
Unstaged changes after reset
M    README.md
```

Only a section of a file

```
$ git reset -p README.md
...
## Pushes
+## Branches
...
Unstage this hunk [y,n,q,a,d,/,s,e,?]?
```

Keep your changes at hand

git stash is your friend

```
$ git stash save "Fix a typo"
$ //git stash save -u "Fix a typo" (also stash untracked files)
Unstaged changes after reset
M    README.md

$ git stash list
stash@{0}: On master: Fix a typo
stash@{1}: On my-branch: Add italian translations
stash@{2}: On master: Fix bug #123

$ git stash show [stash@{0}]
 README.md | 5 +++--
```

Get your changes back

Apply a stash

```
$ git stash apply stash@{1}
...
Changes not staged for commit:
     modified:  README.md
...
```

Apply and drop a stash

```
$ git stash pop stash@{1}
...
Changes not staged for commit:
     modified:  README.md
...
```

And the untracked files

There is a shortcut for that

```
$ git stash -u
Saved working directory and index state WIP on branch-name:
8ad48aa My commit comment
HEAD is now at 8ad48aa My commit comment
```

Remove untracked files or directory

Untracked files

```
$ git clean
```

Untracked directories

```
$ git clean -d
```

Not sure of what you're doing ?

```
$ git clean -n
// equivalent to --dry-run
```

Force

```
$ git clean -f
```

Finally, commit your changes

```
$ git commit -m "An intelligible message" README.md
[master 12345] An intelligible message
  1 file changed, 1 insertion(+), 1 deletion(-)
```

Forgot a file ?

```
$ git add CHANGELOG-1.0.md
$ git commit --amend --no-edit
[master 67890] An intelligible message
  2 files changed, 2 insertion(+), 1 deletion(-)
```

Want to change the message ?

```
$ git commit --amend -m "A new message without typos"
```

Error in a commit...

Cancel a commit
(but do not revert the changes)

```
$ git reset HEAD~1
// 1 is the number of commit to cancel
Unstaged changes after reset:
M    my-updated-file.md
```

Great pushes

Push one branch at the same time

Push the current branch if it's tracked and distant and local name are identicall *(default mode since git 2.0)*

```
$ git config --global push.default simple
```

Push the current branch if it's tracked (regardless of its name)

```
$ git config --global push.default upstream
```

***Warning**: before git 2.0 default mode was matching. Push all branches having the same distant and local name*

First push

Push and track

```
(tracked) $ git push -u origin tracked
...
To git@github.com:antfroger/git-intro.
 * [new branch]  tracked -> tracked

Branch tracked set up to track remote branch to tracked origin
```

Forget to track ?

```
(untracked) $ git branch --set-upstream-to=origin/untracked untracked
Branch untracked set up to track remote branch to untracked origin
```

Clean the history

Before pushing, clean the history: remove useless commit, merge two commits, reorganise commits...

git rebase -i

```
(BUG-123) $ git rebase -i origin/BUG-123
```

```
pick 6e70bd9 Add description
pick 1b6d5ae Add dependency
pick d590486 Forget dependency
pick 08e6ace Fix typo
pick af15bf3 Add changelog + create .gitignore
pick f1f53a9 Create documentation
...
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = se commit, but stop for amending
...
```

Pull correctly

By default, when you pull, git execute a **merge** after pulling.

But what you want is not merging the distant branch into your current branch but get the distant changes and re-execute your changes on the up-to-date local branch = **rebase**.

Merge should only be used to integrate your changes into a parent branch.

```
$ git config --global pull.rebase preserve
```

Merge or rebase ?

What's the difference ?

`Git merge`

Join two or more development histories together

*“ A git merge should only be used for **incorporating the entire feature set of branch into another one** , in order to preserve a useful, semantically correct history graph. Such a clean graph has significant added value.*

Christophe Porteneuve

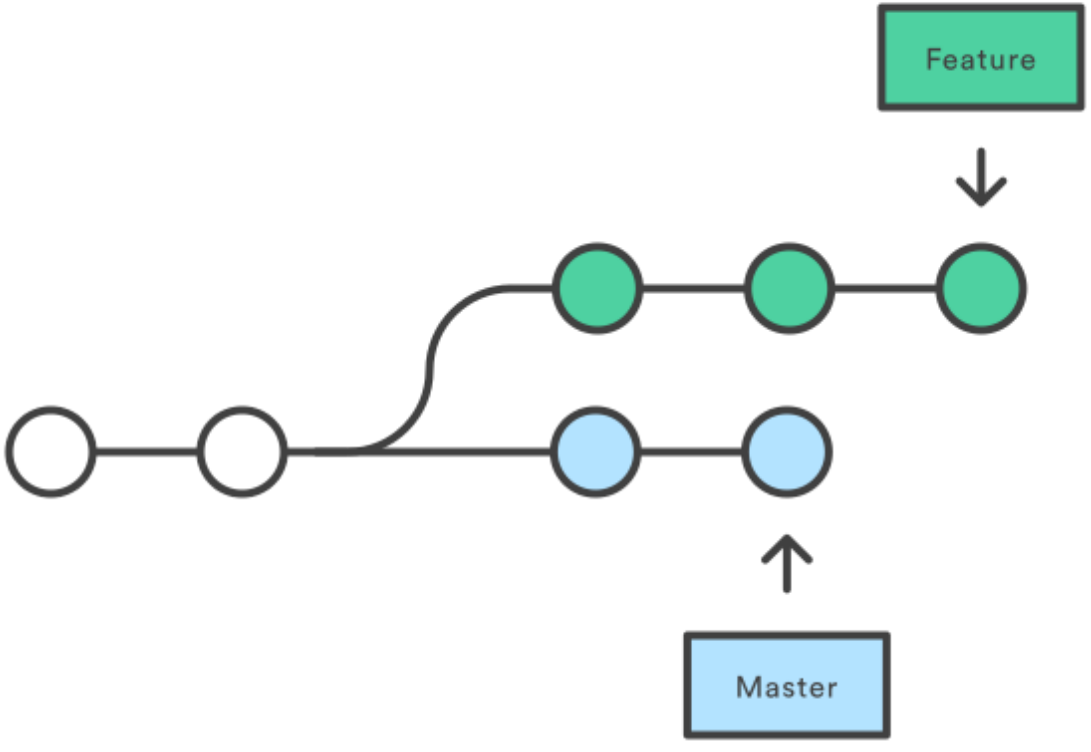
`Git rebase`

Reapply commits on top of another base tip

Example

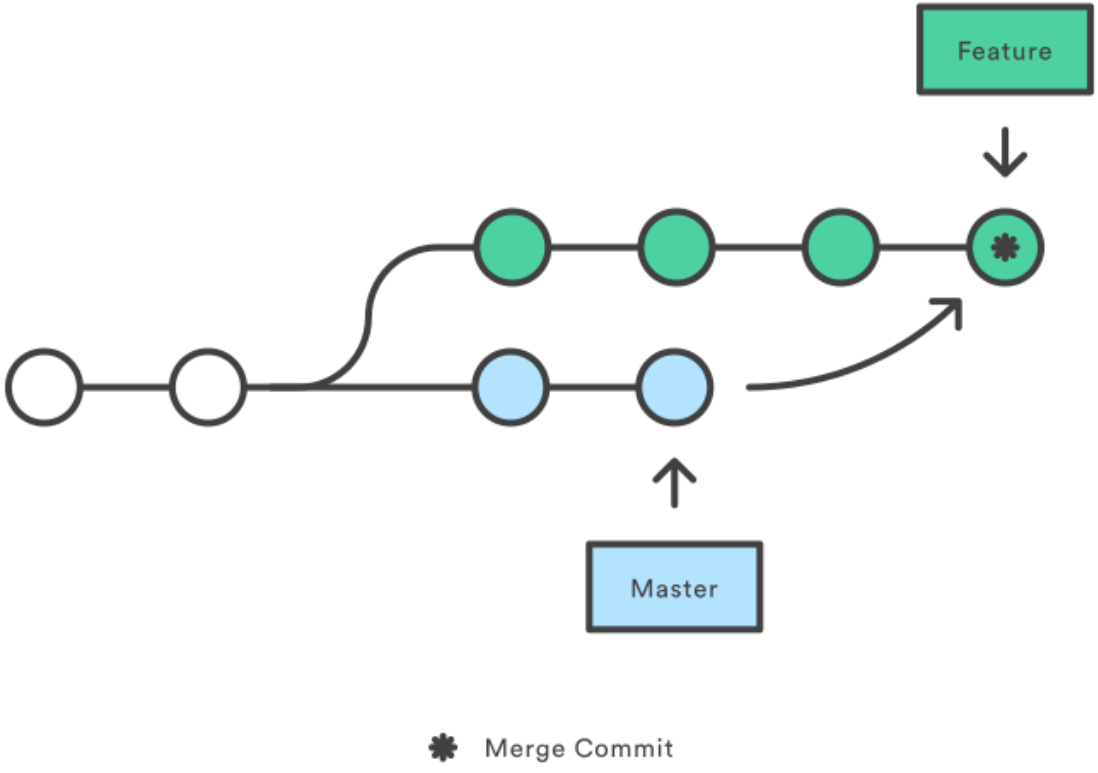
Initial situation

A forked commit history



Merge

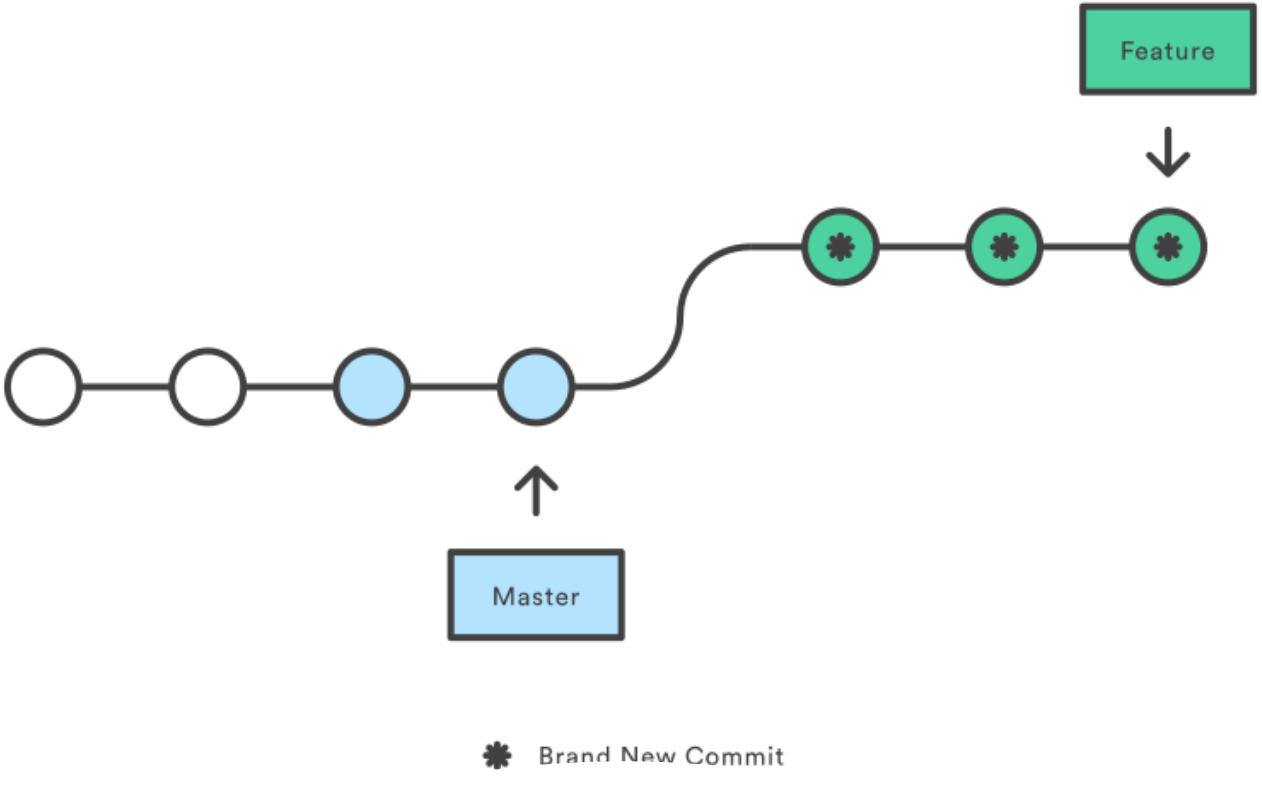
Merging master into the feature branch



Merge Commit

Rebase

Rebasing the feature branch onto master



Others features...

Logs

Logs are useful to understand what was done.

```
$ git log
```

```
commit af6297d608b876e58eaced77479d630df6b3f52a
Author: Antoine Froger <antoine@alittlemarket.fr>
Date:   Fri Jun 17 13:53:40 2016 +0200

    Revert "Create CHANGELOG"

    This reverts commit 5387a753c25483cb4e898c9f3dc65e523e0589c5.

commit 5387a753c25483cb4e898c9f3dc65e523e0589c5
Author: Antoine Froger <antoine@alittlemarket.fr>
Date:   Fri Jun 17 13:53:30 2016 +0200

    Create CHANGELOG

commit 848f919ae81a1579f1c0246cbdd9ddc324e4ce23
Author: Antoine Froger <antoine@alittlemarket.fr>
Date:   Fri Jun 17 13:50:08 2016 +0200

    Add monolog dependency
```

Logs

Options are available

```
$ git log --stat
```

```
commit af6297d608b876e58eaced77479d630df6b3f52a
Author: Antoine Froger <antoine@alittlemarket.fr>
Date:   Fri Jun 17 13:53:40 2016 +0200

    Revert "Create CHANGELOG"

    This reverts commit 5387a753c25483cb4e898c9f3dc65e523e0589c5.

 CHANGELOG.md | 8 --------
 1 file changed, 8 deletions(-)
```

```
$ git log --graph
```

```
| * | commit dfaabc8ffbff69fd922d85f8fc871c27193fbf5d
| | | Author: alittleceline <celine@alittlemarket.fr>
| | | Date:   Mon Jun 13 17:05:03 2016 +0200
| | |
| | |     BX-1119-grid: asset group was renamed in the meanwhile
| | |
| * | commit 779c2f2d29070676984db08abb9092f2becf4a4d
|/ /  Author: alittleceline <celine@alittlemarket.fr>
| |   Date:   Mon Jun 13 13:49:47 2016 +0200
| |
| |       BX-1119-grid: call flexbox grid from fwkcss
| |
* |   commit 4f0842e8b411515f0aed0085a82a71d0d4d322a1
|\ \  Merge: d42936c 0acabb6
| | | Author: Romain Ardiet <romaina@alittlemarket.fr>
| | | Date:   Mon Jun 13 16:32:27 2016 +0200
| | |
```

Logs

And many others...

```
$ git log -p
$ git log -s
$ git log -U
$ git log --raw
$ git log --histogram
$ git log --shortstat
$ git log --summary
$ git log -z
$ git log --name-only
$ git log --check
...
```

La doc de git log: git-scm.com/docs/git-log

Grep

You want to search only on the tracked files ?
git grep is for you

```
$ git grep -a "doc"
composer.lock:                  "doctrine/couchdb": "~1.0@dev",
composer.lock:                  "doctrine/dbal": "~2.2",
composer.lock:                  "doctrine/annotations": "~1.0",
composer.lock:                  "doctrine/common": "~2.2",
...
(END)
```

And more !

Lots of other functions exist
Read the doc !

`git-scm.com/docs`

Thanks

for listening to me

Antoine Froger
github.com/antfroger
@__tooni__