1. SVN > GIT

Changer votre manière de versionner

Jour 1 / Matin

1. Présentation de GIT
2. GIT vs SVN : différences fondamentales (Retour d’expérience)
3. Commandes basiques de GIT (TP - configurer et utiliser GIT)
4. Bonnes pratiques et conventions de GIT

1. Présentation de GIT

- 1991 - 2002 : Linux est maintenu avec des diff et des patch

- 2002 : Le noyau Linux utilise BitKeeper (propriétaire)

- 7 avril 2005 : v 0.0.1 => Rupture Linux / Sté BitKeeper
  Linus Torvalds développe son propre DVCS (Distributed Version Control System)
  peu de temps APRÈS **Bazaar** (bzr - 26 mars 2005), mais AVANT **Mercurial** (hg - 19 avril 2005)

- Développé en : C, Shell Unix, Perl, Tcl et GNU Bash

1. Présentation de GIT

Objectifs de GIT

1. vitesse *(dépôt local / compression)*
2. conception simple 💩
3. support pour les développements non linéaires *(milliers de branches //)*
4. complètement distribué
5. capacité à gérer efficacement des projets d'envergure
   tels que le noyau Linux (vitesse et compacité des données)

1. Présentation de GIT

SVN

- 2000 : Société CollabNet
- 2010 : Apache Fondation
- Développé en : C

1. Présentation de GIT

Les autres VCS Libres

L

C/S

D

1982 : GNU RCS - C - Revision Control System / Projet GNU (**Richard Stallman**)

1990 : CVS - C - Concurrent Version System TEAM => 1.11.23 - 2008

2000 : Subversion - C - Fondation apache

2001 : GNU arch - C - Projet GNU => migré sur Bazaar depuis 2005

2002 : Darcs - Haskell (**commutation des patch**)

2003 : SVK - Perl (surcouche de SVN / mirroring)

2005 : Bazaar - Python, C - Canonical (Ubuntu) & Projet GNU

2005 : Git - C, Shell, Perl, Tcl - Linus Torvald

2005 : Mercurial - Python, C - Matt Mackall (FB)

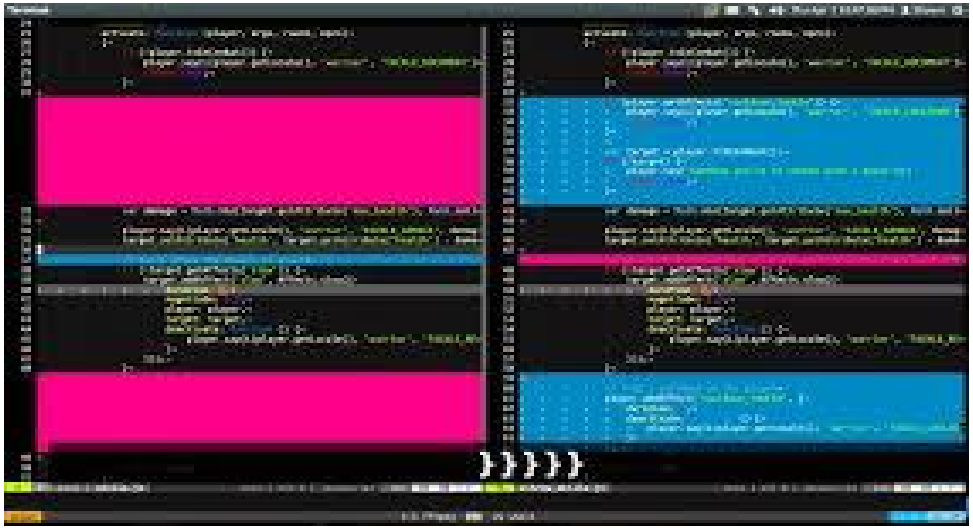
2007 : Fossil - C - D. Richard Hipp (SQLite)

2011 : Veracity - C - SourceGear LLC

} WWW

2. GIT vs SVN

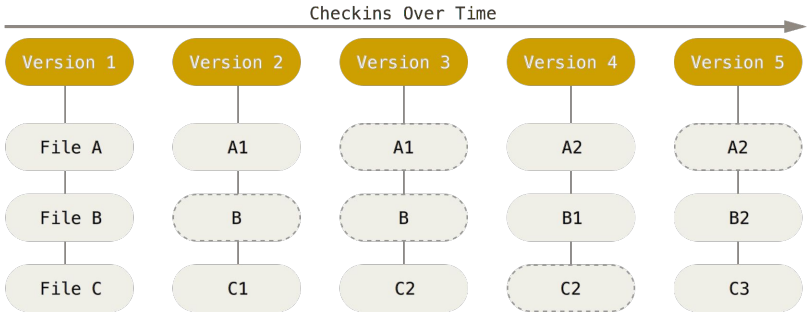
- différences fondamentales
- Retour d’expérience

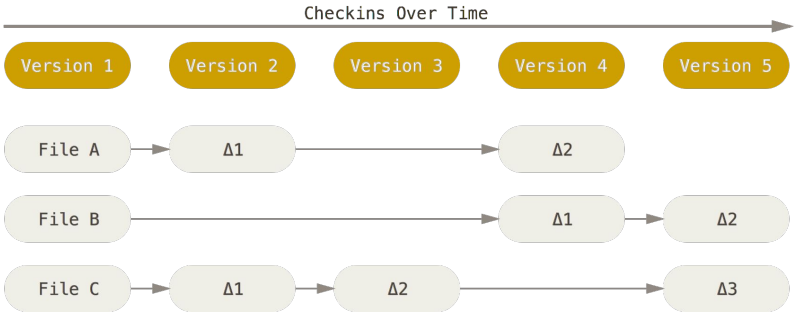


2. GIT vs SVN

1. différences fondamentales

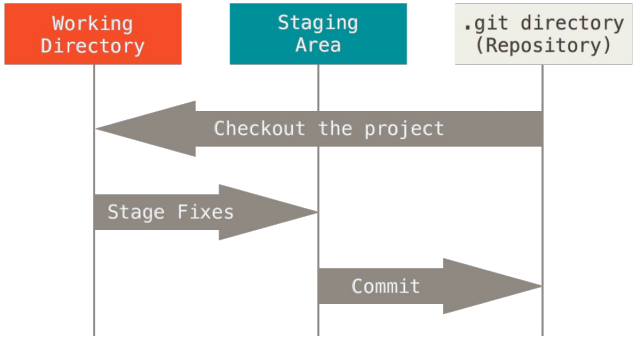
Des instantanés, pas des différences

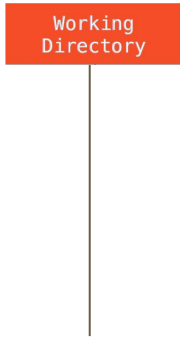




2. GIT vs SVN

Presque toutes les opérations sont locales

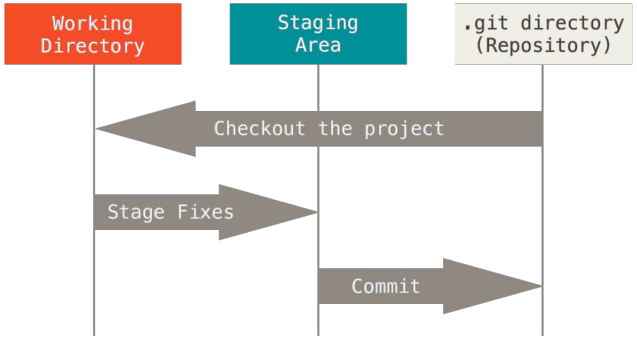


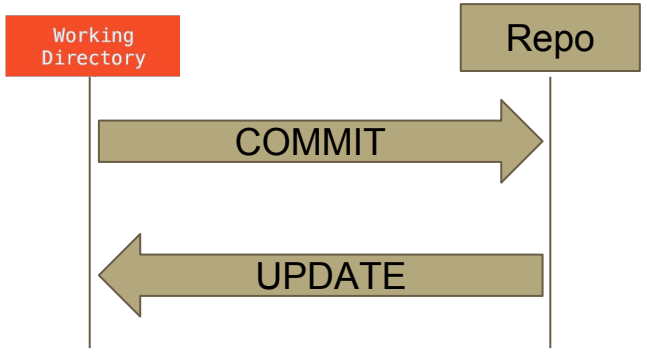


2. GIT vs SVN

2. différences fondamentales

Pas d'intermédiaires entre **Svn** WC & **Svn** Repo

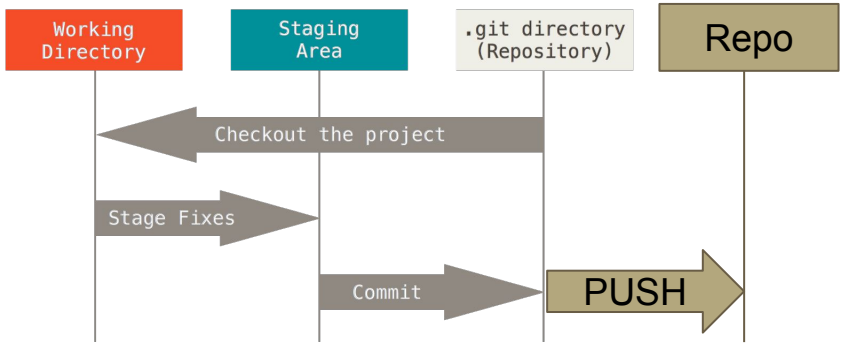


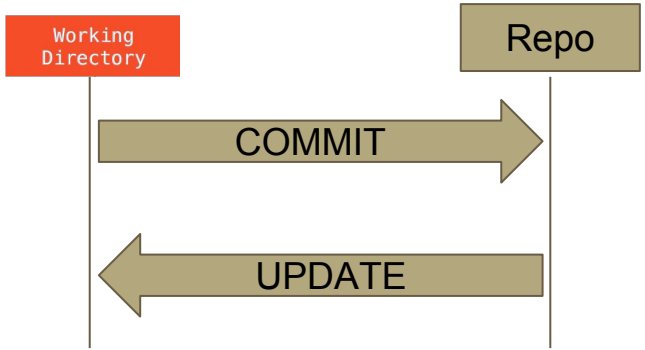


2. GIT vs SVN

2. différences fondamentales

Dans **Git**, il y'a un Repo en **local** ! (et un remote)

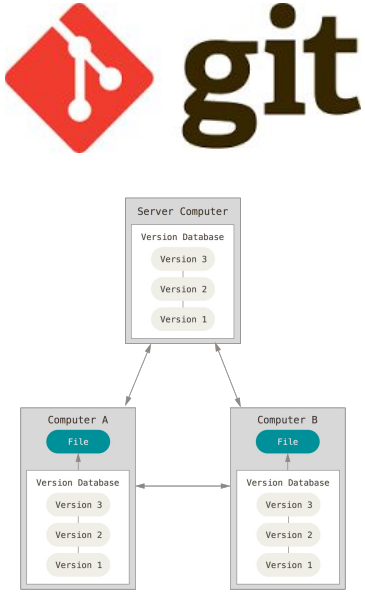


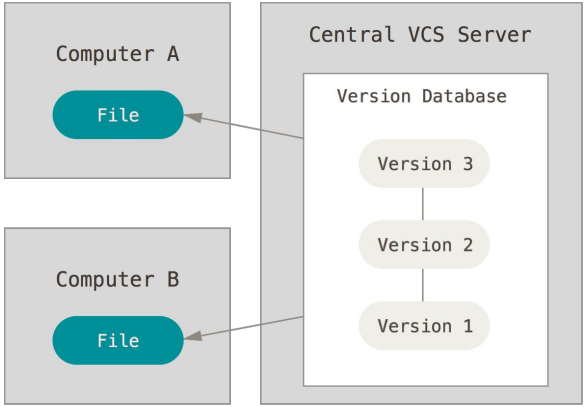


2. GIT vs SVN

3. différences fondamentales

Système distribué != Centralisé (client / serveur)



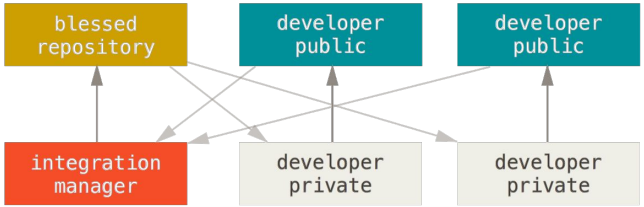


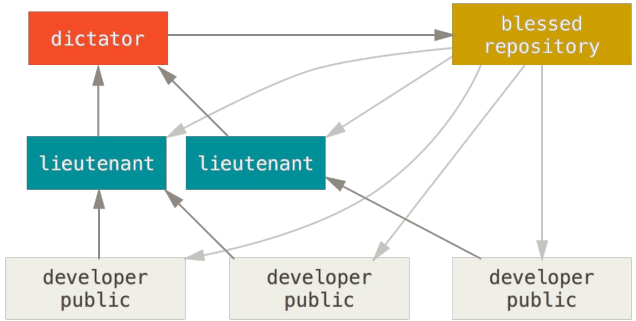
2. GIT vs SVN

4. différences fondamentales

Multi-Dépôt

(SVK)





2. GIT vs SVN

Git gère l'intégrité


```
~$  git log-sha-only
* 12a020088fdc936f5d796d89aa4e2e88d7156471
* f9334a5ac4a93bd6734955abce069d0bced0c064
* 2608b961fa1bfd80cfe0734789b227425832c1ee
* 34fe5e8e73a76f70dbd13ad421f1a72f4f92506d
* 2a6ec9aada17a2bbbd466133e311a95cdcc4438b
* 374f6ee9132e278a339e0b642f3f000722c0835c
* 1c38b3dc4932a78ad8725959a0e158e5817e7a1f
* bafc58aeb94714b6537975272dbd160eb2c15a42
```

```
~$ svn log -q
r24 | sylvain | 2016-04-13 19:07:59
r23 | sylvain | 2016-04-13 19:07:59
r22 | sylvain | 2016-04-13 19:07:59
r21 | sylvain | 2016-04-13 19:07:59
r20 | sylvain | 2016-04-13 19:07:59
r19 | sylvain | 2016-04-13 18:51:07
r18 | sylvain | 2016-04-13 18:51:07
r17 | sylvain | 2016-04-13 18:51:07
r16 | sylvain | 2016-04-13 12:29:21
```

2. GIT vs SVN

En Résumé : différences fondamentales

1. Snapshots
2. Repo Local à 3 états
3. Système distribué
4. Multi-Dépôt
5. Intégrité - SHA1

1. Delta (diff)
2. Repo Distant
3. Système centralisé
4. *(svk)*
5. Révisions (1, 2, 3)

2. GIT vs SVN

Retour d'expérience

- Problématiques rencontrées lors de la prise en main de GIT ?
- Concepts pas ou mal compris ?
- …
- Nous reviendrons plus en détail sur les notions :
  - staging area (cycle de vie d'un fichier)
  - Blob, Tree, Commit, Tag (branches et pointeurs)

3. Commandes basiques de GIT

Travaux Pratiques

- configurer son environnement
  - git config [--local]
  - ~/.gitconfig & .git/config
- Utiliser GIT en local
  - init / status
  - add / rm
  - commit
  - log
  - .gitignore
  - checkout, reset
  - revert
- Branches et pointeurs
  - branch
  - tag
  - merge / rebase
- Utiliser les commandes distantes
  - remote
  - clone
  - push
  - pull
  - fetch

3. Commandes basiques de GIT

TP : git config et help

- git config --system
- git config --global
- git config

- /etc/gitconfig
- ~/.gitconfig
- .git/config

Identité

- git config --global user.name "John Doe"
- git config --global user.email "j@foo.bar"
- git config --global core.editor vim

```
[user]
  name          = John Doe
  email         = j@foo.bar
[core]
  editor        = vim
```

3. Commandes basiques de GIT

TP : git config et help

Consultation

- git config --list
- git config user.name

```
user.name=Sylvain Just
user.email=sy.just@i-sloth.com
core.editor=vim
merge.tool=vimdiff
alias.pop=stash pop
....
```

Aide => man page

- git help <cmd>
- git <cmd> --help
- man git-<cmd>

```
~$ git help -a list all <cmd> (>150)
~$ git help -g guides (tutorials)
attributes   Defining attributes per path
glossary     A Git glossary
ignore       Specifies intentionally …

~$ git help ignore
```

3. Commandes basiques de GIT

TP : git init / status / add

```
~$ git init git_wc && cd git_wc
~$ git status
~$ echo "first file" > file.txt
~$ echo "second file" > f2.txt
~$ git status

~$ git add *.txt
~$ git status

~$ git status --short


~$ git commit -m "first commit"
~$ git status
```

On branch master, nothing to commit

Untracked files:

f2.txt
file.txt

Changes to be committed:

new file: f2.txt
new file: file.txt

A f2.txt
A file.txt

nothing to commit, working directory clean

3. Commandes basiques de GIT

TP : git init / status / add

```
~$ git init git_wc && cd git_wc
~$ git status --short
~$ echo "first file" > file.txt
~$ echo "second file" > f2.txt
~$ git status

~$ git add *.txt
~$ git status --short




~$ git commit -m "first commit"
~$ git status --short
```

```
?? f2.txt
?? file.txt

A  f2.txt
A  file.txt
```

3. Commandes basiques de GIT

TP : git rm / commit

```
~$ echo "foo@mc.com" >> file.txt
~$ git add file.txt
~$ git commit -m "email here"

~$ rm f2.txt
~$ git rm f2.txt

~$ git commit -m "remove f2"
```

2ème commit

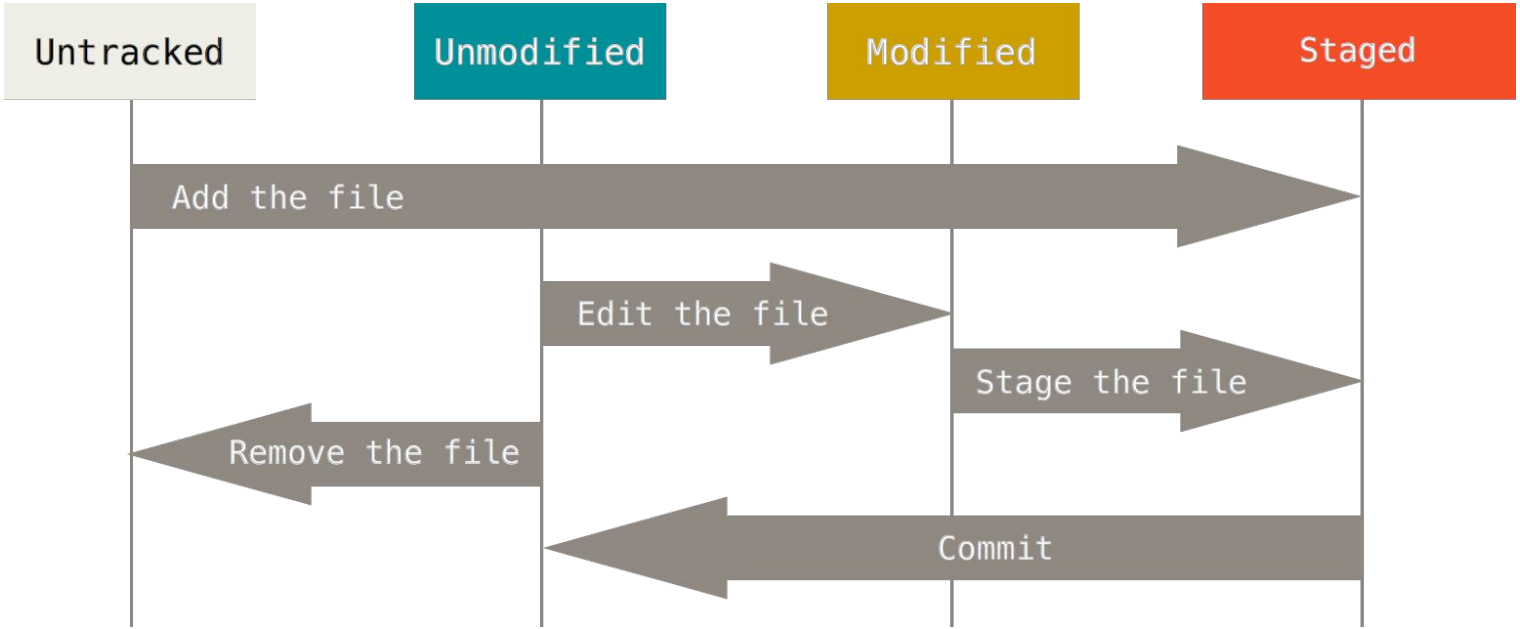
git status --short

D f2.txt
D f2.txt

3ème commit

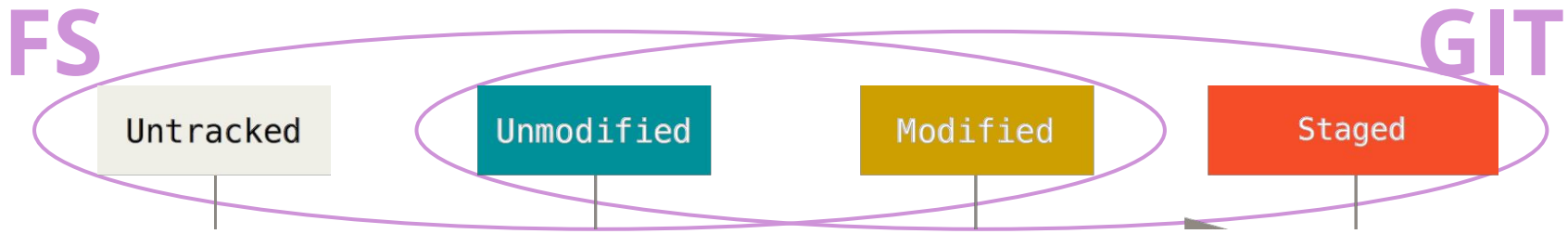
3. Commandes basiques de GIT

TP : git status



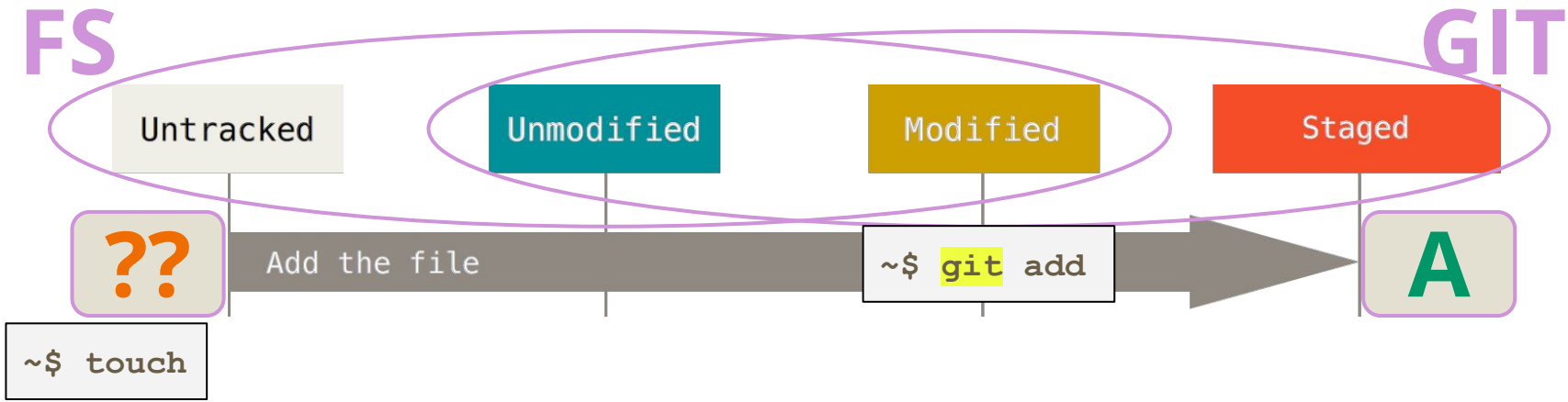
3. Commandes basiques de GIT

TP : git status



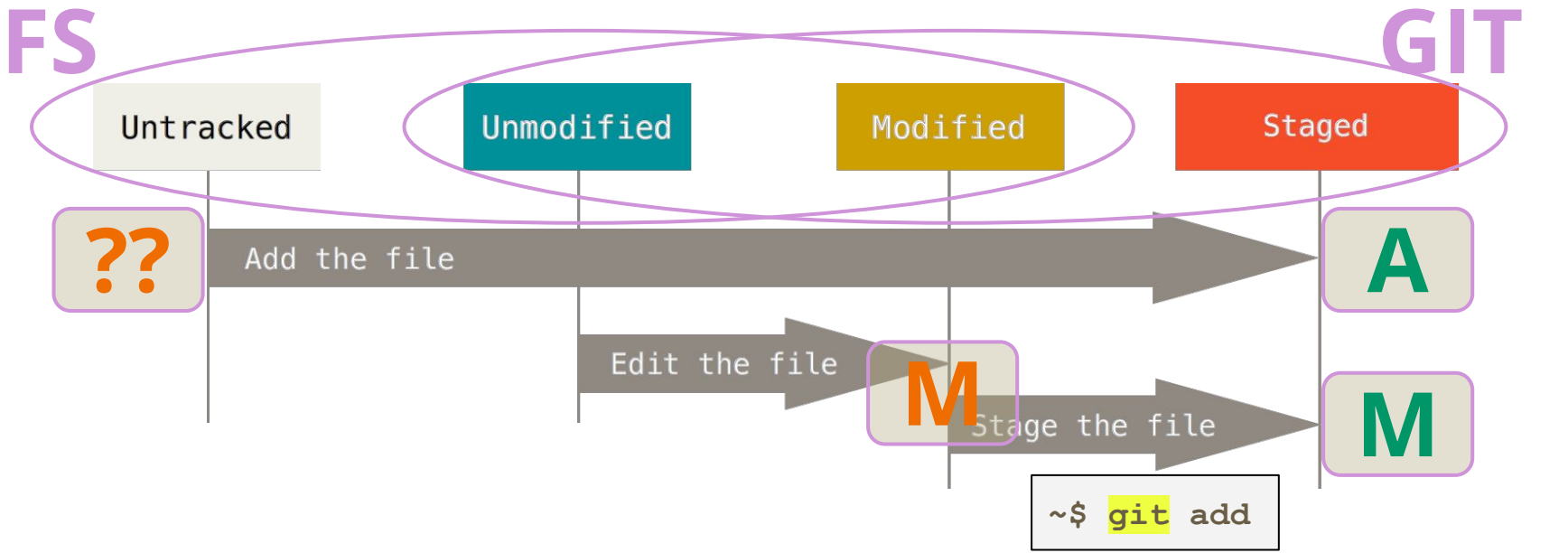
3. Commandes basiques de GIT

TP : git status



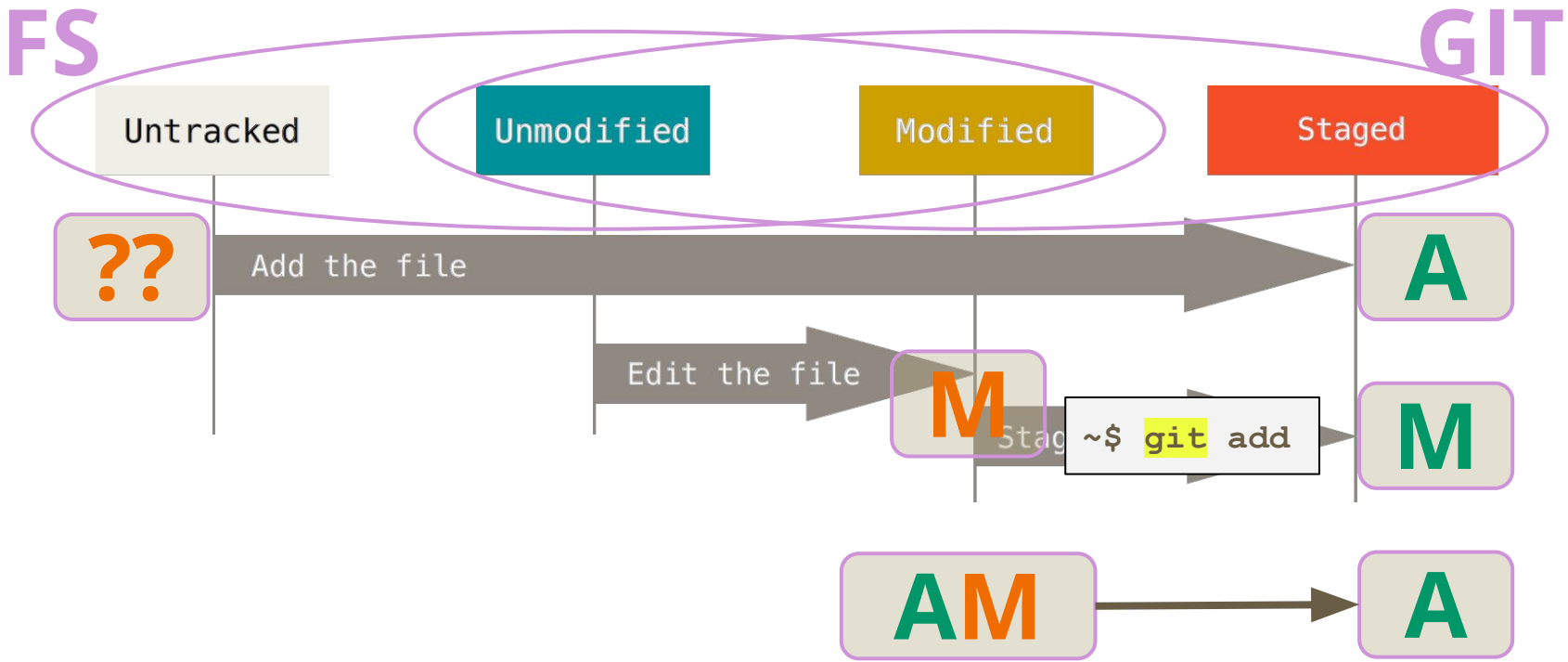
3. Commandes basiques de GIT

TP : git status



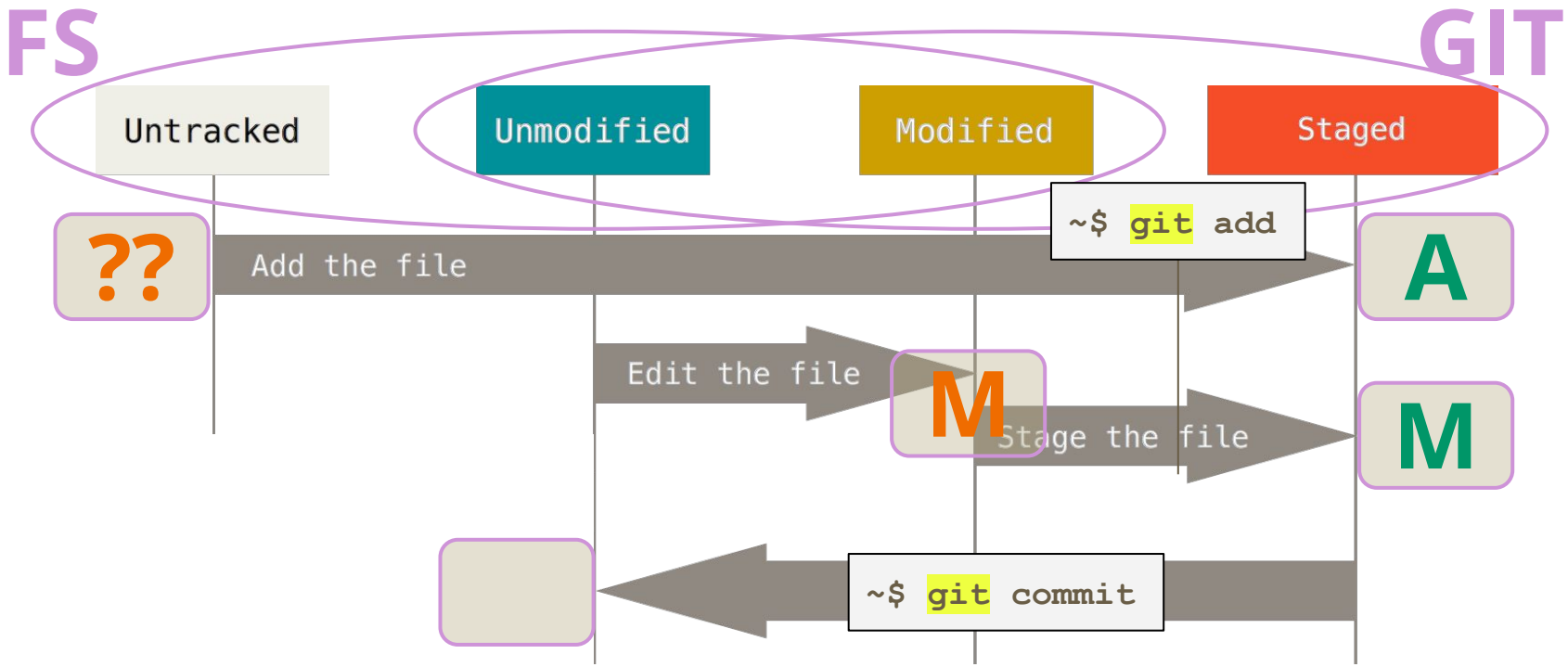
3. Commandes basiques de GIT

TP : git status



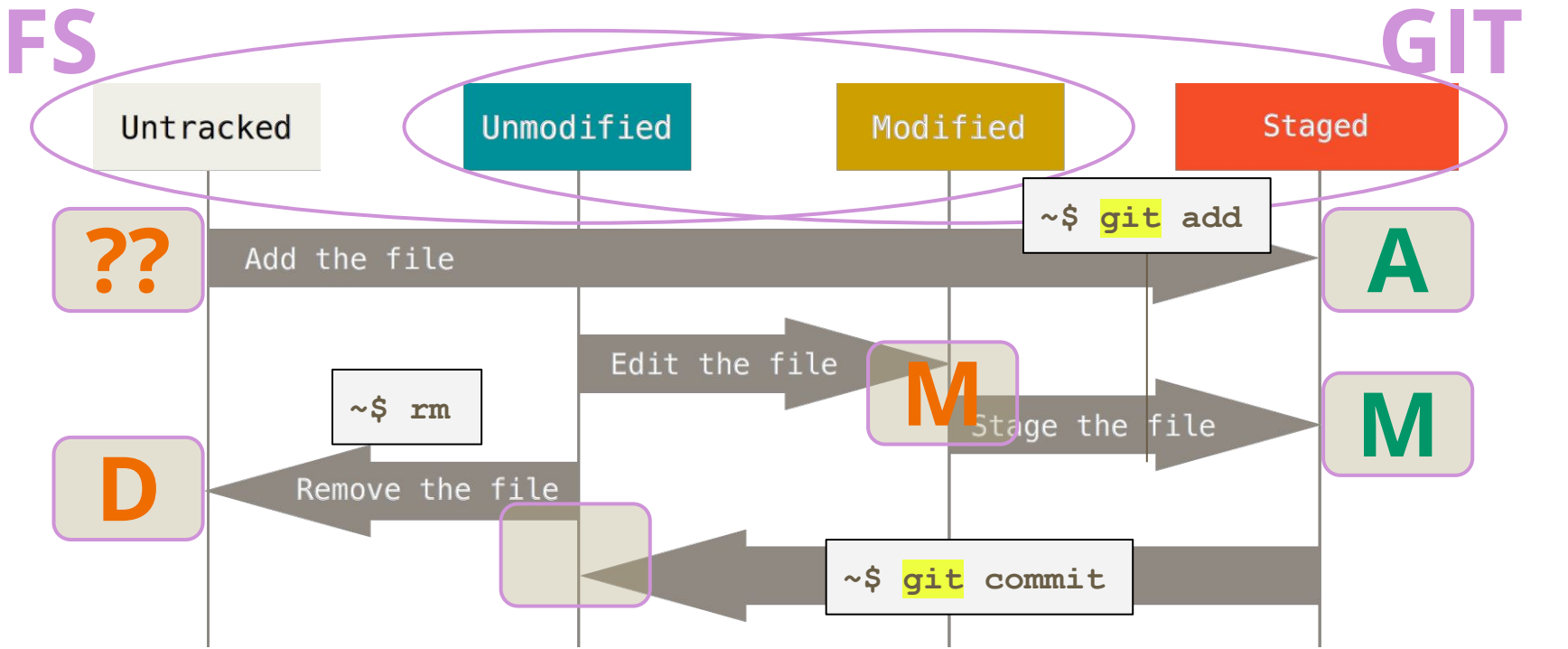
3. Commandes basiques de GIT

TP : git status



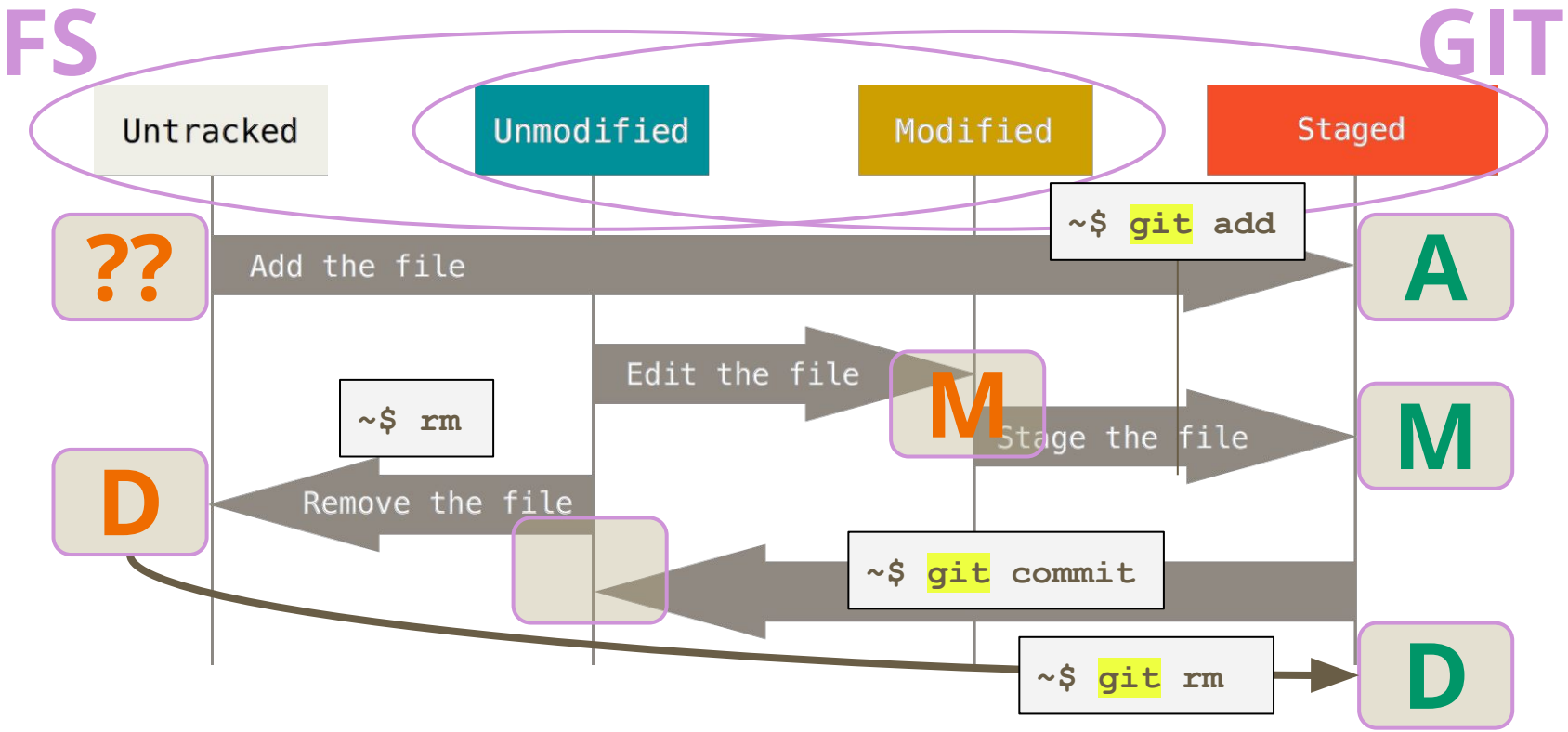
3. Commandes basiques de GIT

TP : git status



3. Commandes basiques de GIT

TP : git status



3. Commandes basiques de GIT

TP : git status

Staged

A	Added
M	Modified
D	Deleted

Unstaged

??	Untracked
M	Modified
D	Deleted

3. Commandes basiques de GIT

TP : git status

```
X          Y     Meaning
-------------------------------------------------
           [MD]   not updated
M        [ MD]   updated in index
A        [ MD]   added to index
D         [ M]   deleted from index
R        [ MD]   renamed in index
C        [ MD]   copied in index
[MARC]           index and work tree matches
[ MARC]     M    work tree changed since index
[ MARC]     D    deleted in work tree
```

```
X          Y     Meaning
-------------------------------------------------
D          D     unmerged, both deleted
A          U     unmerged, added by us
U          D     unmerged, deleted by them
U          A     unmerged, added by them
D          U     unmerged, deleted by us
A          A     unmerged, both added
U          U     unmerged, both modified
-------------------------------------------------
?          ?     untracked
!          !     ignored
-------------------------------------------------
```

3. Commandes basiques de GIT

Travaux Pratiques : git log

options de git log

- -p (patch)
- --stat (+++-----)
- --shortstat (files)
- --abbrev-commit
- --relative-date
- --graph
- --pretty

Limiter la sortie de git log

- -(n)
- --since, --after
- --until, --before
- --author
- --committer

3. Commandes basiques de GIT

TP : git log --pretty

options de format de git log --pretty

- %s Sujet
- %H SHA-1 du **commit**
- %h SHA-1 abrégée du commit
- %T SHA-1 de l'arborescence (**tree**)
- %t SHA-1 abrégée de l'arborescence
- %P SHA-1 des **parents**
- %p SHA-1 abrégées des parents

- %an Nom de l'auteur
- %ae E-mail de l'auteur
- %ad Date de l'auteur (cf -date=)
- %ar Date relative de l'auteur
- %cn Nom du validateur
- %ce E-mail du validateur
- %cd Date du validateur
- %cr Date relative du validateur

```
~$ vi ~/.gitconfig
```

```
~$ vi .git/config
```

3. Commandes basiques de GIT

TP : gitignore

```
~$ ls -la
drwxr-xr-x+ .git/
-rw-r--r--+ file.txt
```

```
drwxr-xr-x+ refs
drwxr-xr-x+ objects
drwxr-xr-x+ info
drwxr-xr-x+ hooks
-rw-r--r--+ description
-rw-r--r--+ config
drwxr-xr-x+ branches
-rw-r--r--+ HEAD

drwxr-xr-x+ logs
-rw-r--r--+ COMMIT_EDITMSG
-rw-r--r--+ index
```

3. Commandes basiques de GIT

TP : gitignore

```
~$ ls -la
drwxr-xr-x+ .git/
-rw-r--r--+ file.txt

~$ touch file.tmp
~$ mkdir -p etc usr/{s,}bin
~$ mkdir etc usr/bin/local
~$ touch etc/file.{cnf,tmp,log}
~$ touch usr/{s,}bin/f.{tmp,log}
~$ touch usr/bin/local/f.tmp
```

```
~$ git status --short
?? etc/
?? file.log
?? file.tmp
?? usr/
```

```
find ./ -type d && touch $i/.keepme
git add * && git commit "keep dirs"

git ls-files -o --exclude-standard
```

3. Commandes basiques de GIT

TP : gitignore

```
~$ ls -la
drwxr-xr-x+ .git/
-rw-r--r--+ file.txt


~$ touch file.tmp
~$ mkdir -p etc usr/{s,}bin
~$ mkdir etc usr/bin/local
~$ touch etc/file.{cnf,tmp,log}
~$ touch usr/{s,}bin/f.{tmp,log}
~$ touch usr/bin/local/f.tmp
```

```
~$ git status --short


?? file.log
?? file.tmp
?? etc/file.cnf
?? etc/file.log
?? etc/file.tmp

?? usr/sbin/file.log
?? usr/sbin/file.tmp

?? usr/bin/file.log
?? usr/bin/file.tmp
?? usr/local/bin/file.log
?? usr/local/bin/file.tmp
```

3. Commandes basiques de GIT

TP : gitignore

```
~$ ls -la
drwxr-xr-x+ .git/
-rw-r--r--+ file.txt

~$ mkdir -p etc usr/{s,}bin
~$ touch usr/{s,}bin/f.{tmp,log}

~$ echo "modif" >> file.txt

~$ echo "/file.*" > .gitignore

~$ echo "*.tmp" >> .gitignore
```

```
~$ git status --short
 M file.txt
?? .gitignore
?? file.log
?? file.tmp
?? etc/file.cnf
?? etc/file.log
?? etc/file.tmp

?? usr/sbin/file.log
?? usr/sbin/file.tmp

?? usr/bin/file.log
?? usr/bin/file.tmp
?? usr/local/bin/file.log
?? usr/local/bin/file.tmp
```

3. Commandes basiques de GIT

TP : gitignore

```
~$ ls -la
drwxr-xr-x+ .git/
-rw-r--r--+ file.txt

~$ mkdir -p etc usr/{s,}bin
~$ touch usr/{s,}bin/f.{tmp,log}

~$ echo "modif" >> file.txt
~$ echo "/file.*" > .gitignore
~$ echo "*.tmp" >> .gitignore

~$ echo "etc/*.log" >> .gitignore

~$ echo "**/bin" >> .gitignore
```

```
~$ git status --short
 M file.txt
?? .gitignore
?? file.log
?? file.tmp
?? etc/file.cnf
?? etc/file.log
?? etc/file.tmp

?? usr/sbin/file.log
?? usr/sbin/file.tmp

?? usr/bin/file.log
?? usr/bin/file.tmp
?? usr/local/bin/file.log
?? usr/local/bin/file.tmp
```

3. Commandes basiques de GIT

TP : gitignore

```
~$ git status --short
 M file.txt
?? .gitignore
?? file.log
?? file.tmp
?? etc/file.cnf
?? etc/file.log
?? etc/file.tmp

?? usr/sbin/file.log
?? usr/sbin/file.tmp

?? usr/bin/file.log
?? usr/bin/file.tmp
?? usr/local/bin/file.log
?? usr/local/bin/file.tmp
```

```
~$ git status --short
 M file.txt
?? .gitignore
?? etc/file.cnf
?? usr/sbin/file.log
```

cat .gitignore

- *.tmp
- /file.*
- etc/*.log
- **/bin

3. Commandes basiques de GIT

TP : git commit, reset, co & rm

Annuler des actions

```
~$ git commit --amend -m "mess"
```

Modifier le message du dernier commit (**/!\ et le SHA-1**)

```
~$ git reset HEAD <modified>
~$ git rm --cached <added>
```

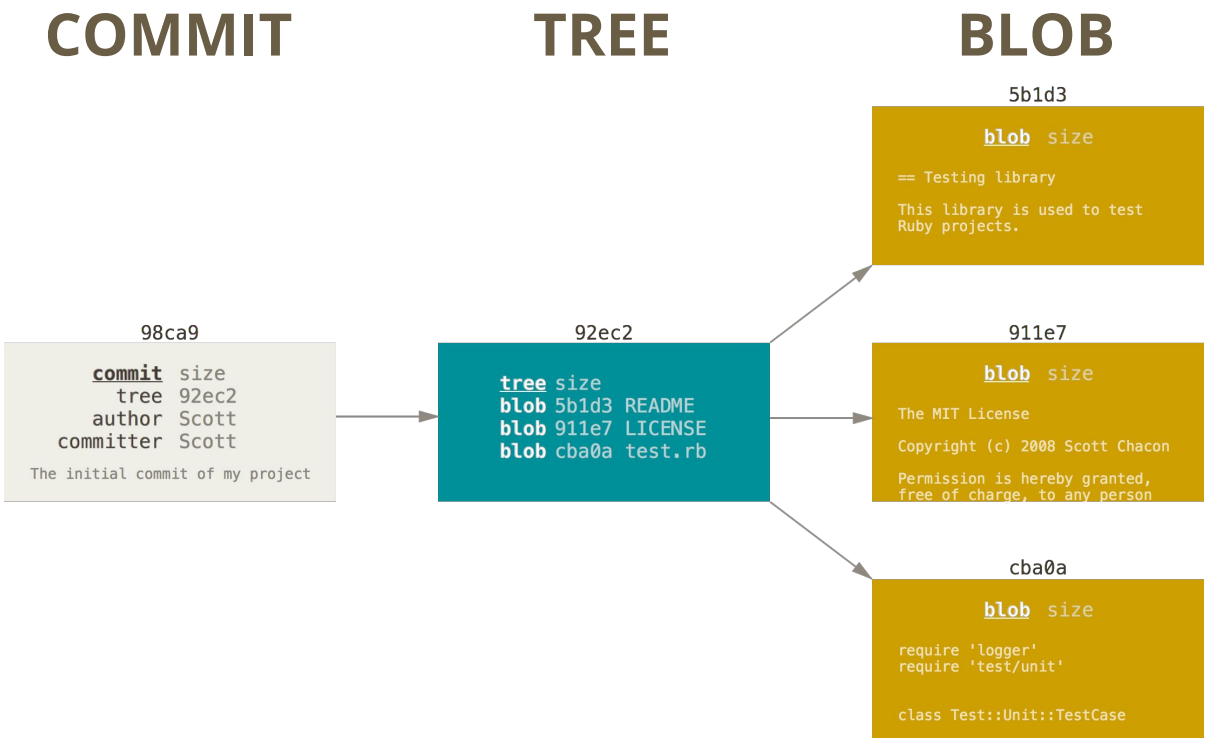
Désindexer un fichier (modifié ou ajouté) / revenir à une version antérieure (du fichier)

```
~$ git checkout <modified>
```

revenir à l'état du dernier commit (**/!\ perte des modifs**)

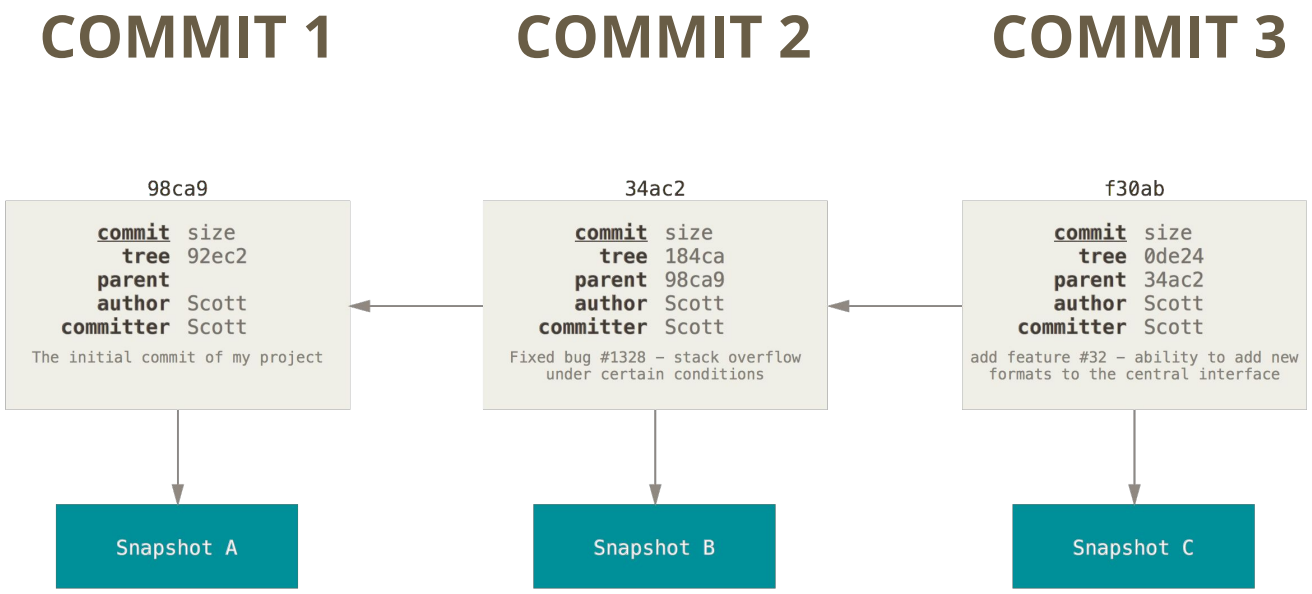
3. Commandes basiques de GIT

TP : git branch (pointeurs)



3. Commandes basiques de GIT

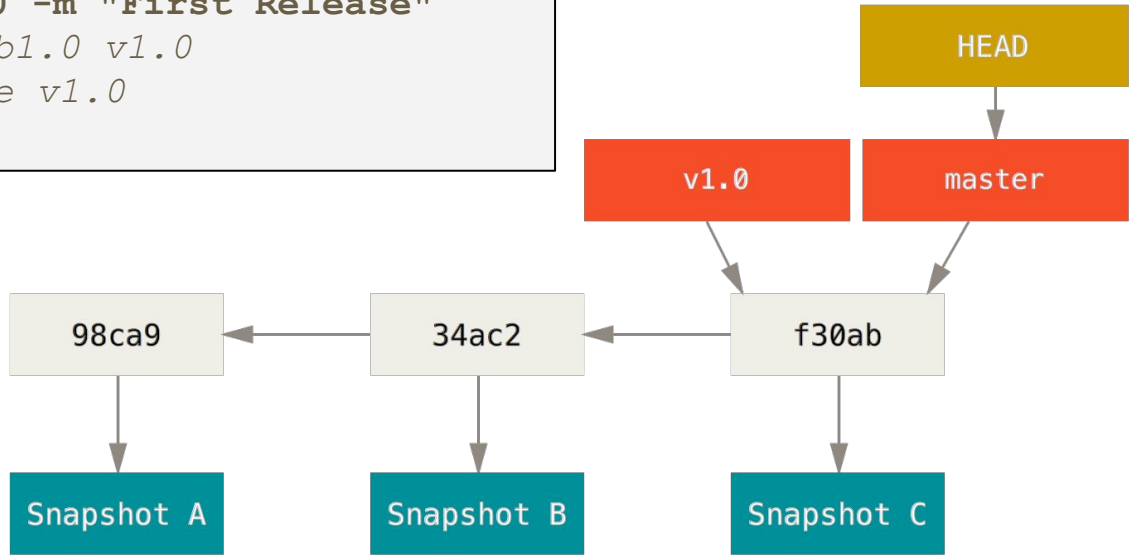
TP : git branch (pointeurs)



3. Commandes basiques de GIT

TP : git branch (pointeurs)

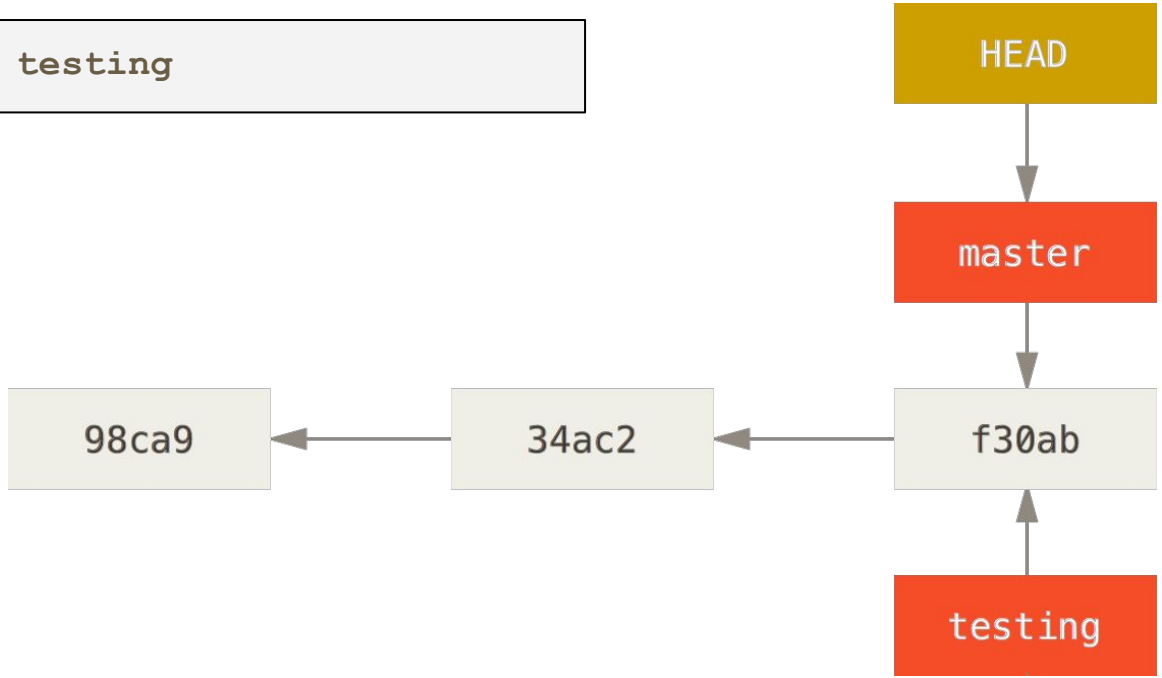
```
~$ git tag v1.0 -m "First Release"
~$ git branch b1.0 v1.0
~$ git describe v1.0
```



3. Commandes basiques de GIT

TP : git branch (pointeurs)

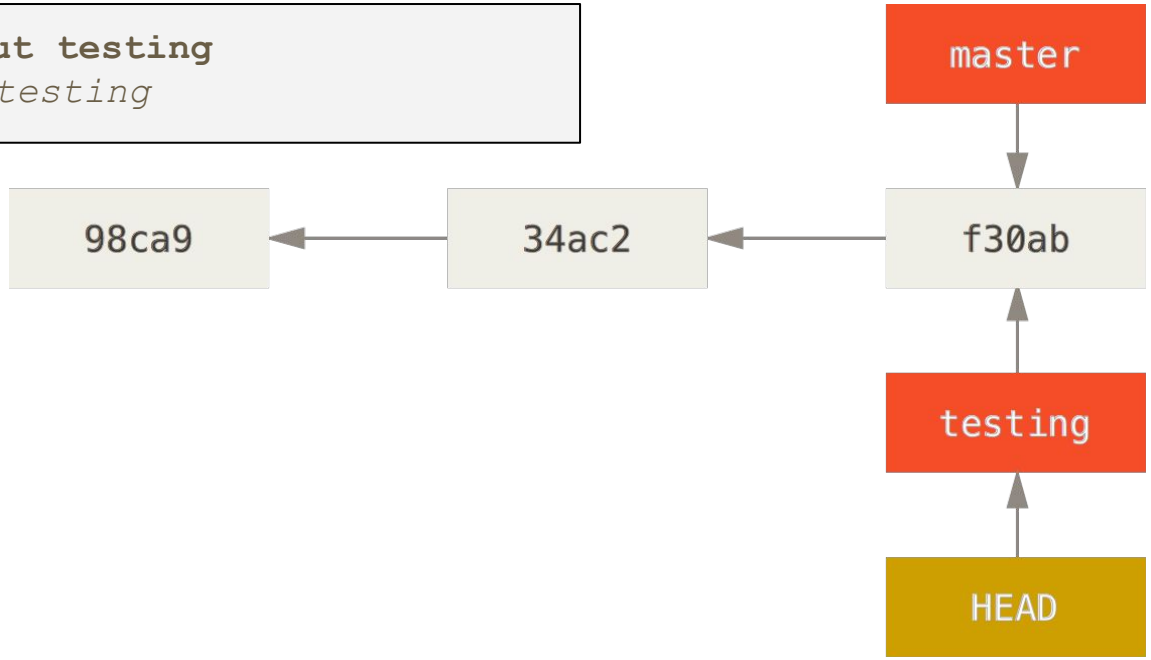
```
~$ git branch testing
```



3. Commandes basiques de GIT

TP : git branch (pointeurs)

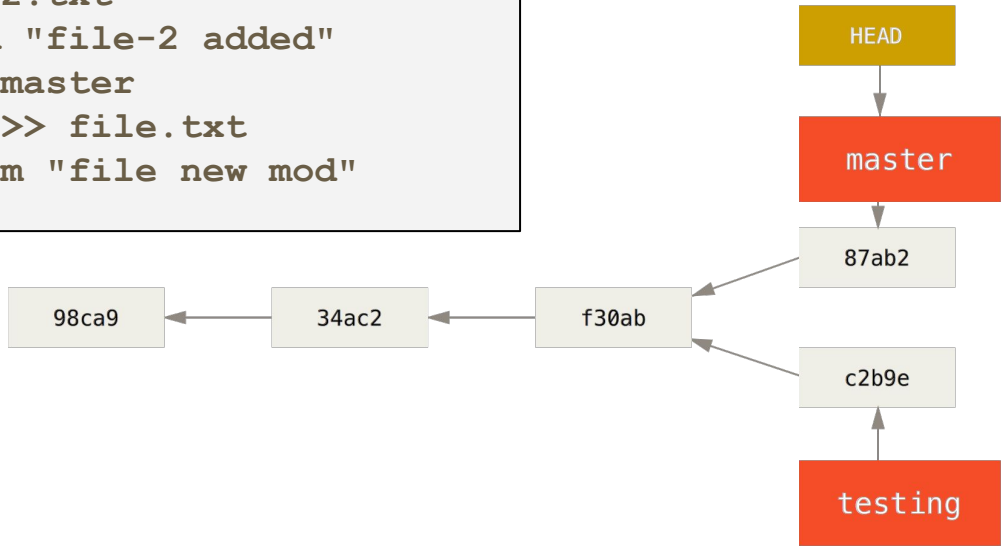
```
~$ git checkout testing
~$ git co -b testing
```



3. Commandes basiques de GIT

TP : git branch (pointeurs)

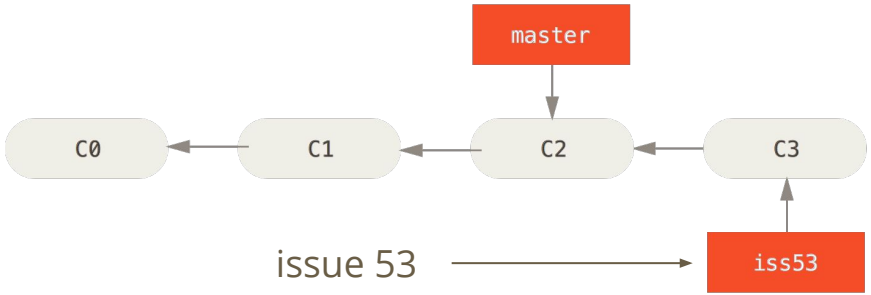
```
~$ touch file-2.txt
~$ git add file-2.txt
~$ git commit -m "file-2 added"
~$ git checkout master
~$ echo "modif" >> file.txt
~$ git commit -am "file new mod"
```



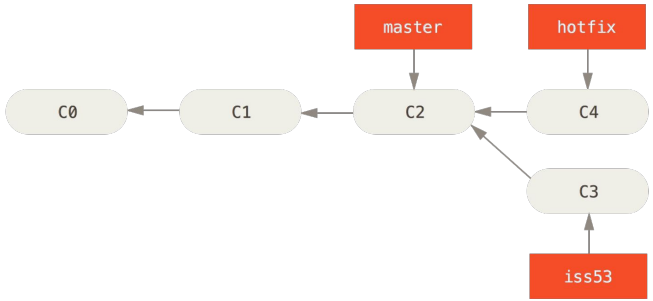
3. Commandes basiques de GIT

TP : git branch (pointeurs)

```
~$ git reset --hard HEAD^
~$ git branch -m testing iss53
```



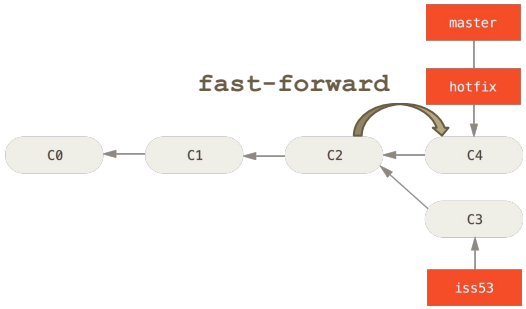

```
~$ git checkout -b hotfix
~$ sed -i 's/foo/bar/' file.txt
~$ git commit -am "mail > bar"
```



3. Commandes basiques de GIT

TP : git branch (pointeurs)

```
~$ git checkout master
~$ git merge hotfix
Updating 4a47382..01a3e99
Fast-forward
 file.txt | 1 +
 1 file changed, 1 insertion(+)
```

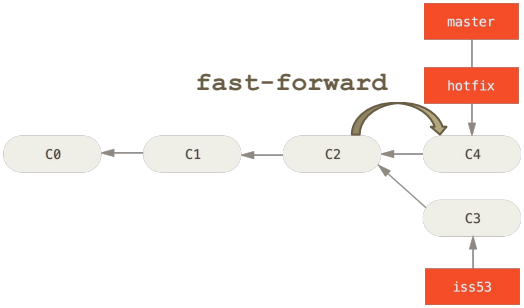


3. Commandes basiques de GIT

TP : git branch (pointeurs)

```
~$ git checkout master
~$ git merge hotfix
Updating 4a47382..01a3e99
Fast-forward
 file.txt | 1 +
 1 file changed, 1 insertion(+)
```

log --stat
--shortstat



git log --graph --all

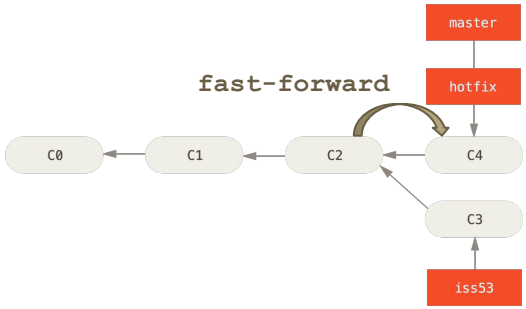
```
* 01a3e99e - (HEAD, master, hotfix) mail > bar on hotfix(7 minutes ago)
| * 2c61ce31 - (iss53) file-2 added (7 minutes ago)
|/
* 4a47382d - (tag: v1.0) remove f2 (3 days ago)
* bd53f07e - email here (3 days ago)
* 7801bc71 - first commit (3 days ago)
```

3. Commandes basiques de GIT

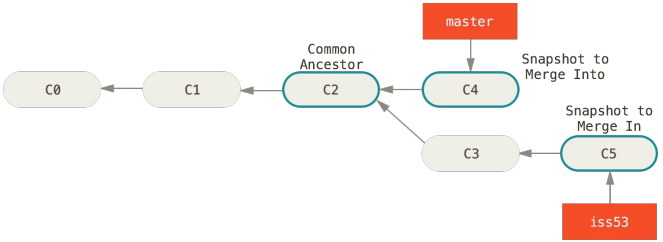
TP : git branch (pointeurs)

```
~$ git checkout master
~$ git merge hotfix
Updating 4a47382..01a3e99
Fast-forward
 file.txt | 1 +
 1 file changed, 1 insertion(+)
```

log --stat
--shortstat

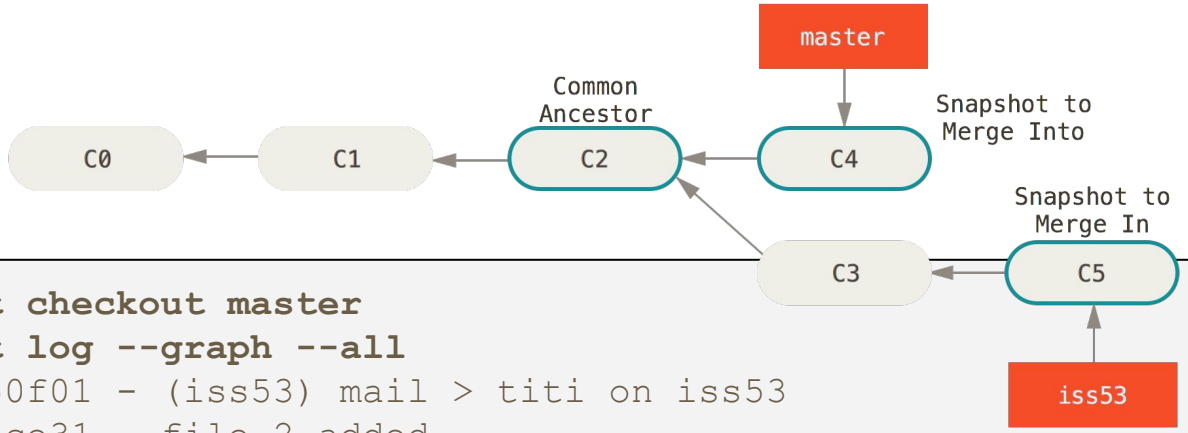



```
~$ git checkout iss53
~$ sed 's/foo/titi/' file.txt
~$ git commit -am "mail > titi on iss53"
```



3. Commandes basiques de GIT

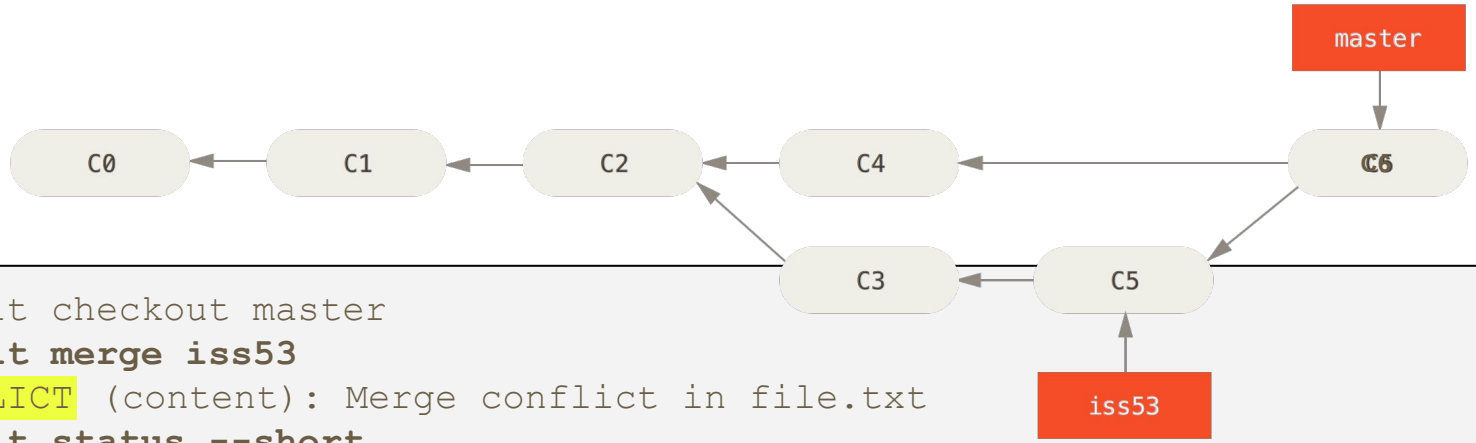
TP : git branch (merge)




```
~$ git checkout master
~$ git log --graph --all
* 8ed50f01 - (iss53) mail > titi on iss53
* 2c61ce31 - file-2 added
| * 86a5a411 - (HEAD, master, hotfix) mail > bar on hotfix
|/
* 4a47382d - (tag: v1.0) remove f2
* bd53f07e - email here (3 days ago)
* 7801bc71 - first commit (3 days ago)
```

3. Commandes basiques de GIT

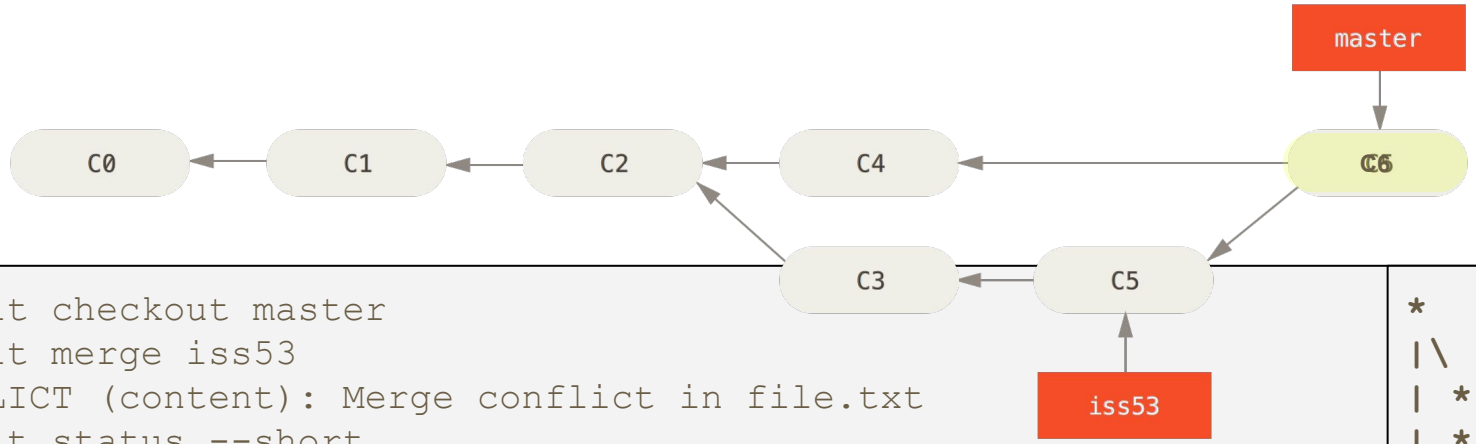
TP : git branch (merge)




```
~$ git checkout master
~$ git merge iss53
CONFLICT (content): Merge conflict in file.txt
~$ git status --short
A  file-2.txt
UU file.txt
~$ git mergetool
~$ git commit
```

3. Commandes basiques de GIT

TP : git branch (merge)

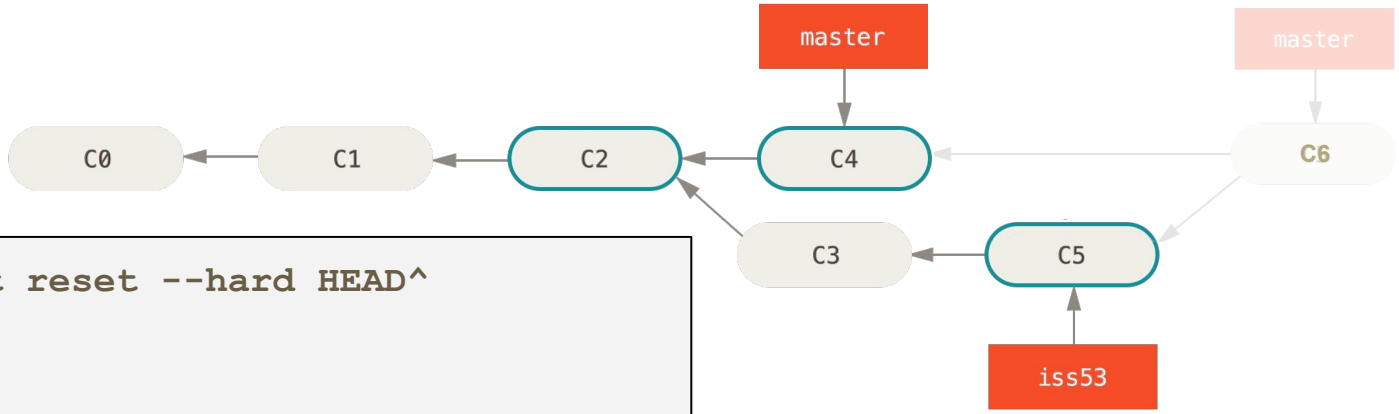



```
~$ git checkout master
~$ git merge iss53
CONFLICT (content): Merge conflict in file.txt
~$ git status --short
A  file-2.txt
UU file.txt
~$ git mergetool
~$ git commit
```

```
*   3bd8fb1c
|\
| * 8ed50f01
| * 2c61ce31
* | 86a5a411
|/
* 4a47382d
* bd53f07e
* 7801bc71
```

3. Commandes basiques de GIT

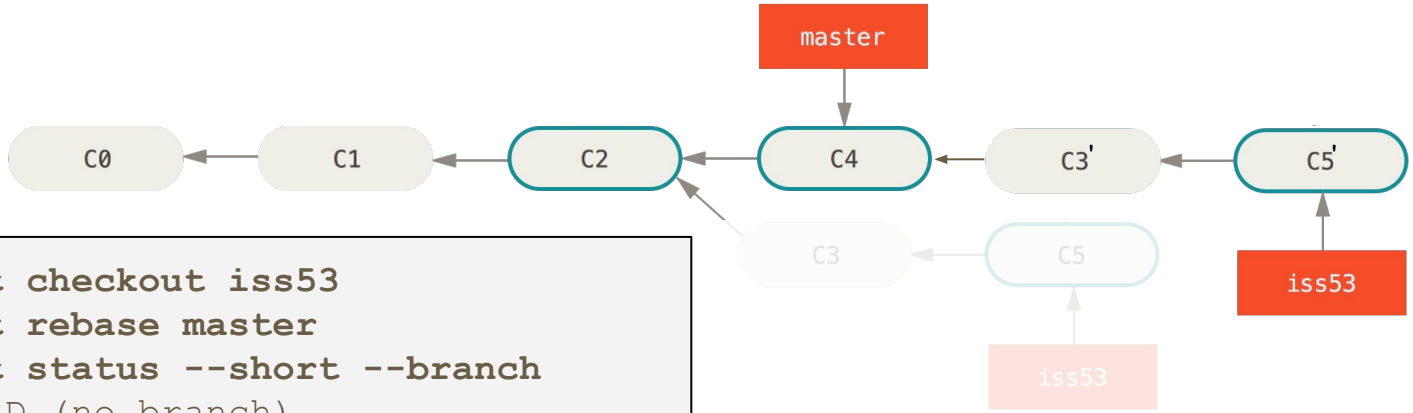
TP : git branch (rebase)




```
~$ git reset --hard HEAD^
```

3. Commandes basiques de GIT

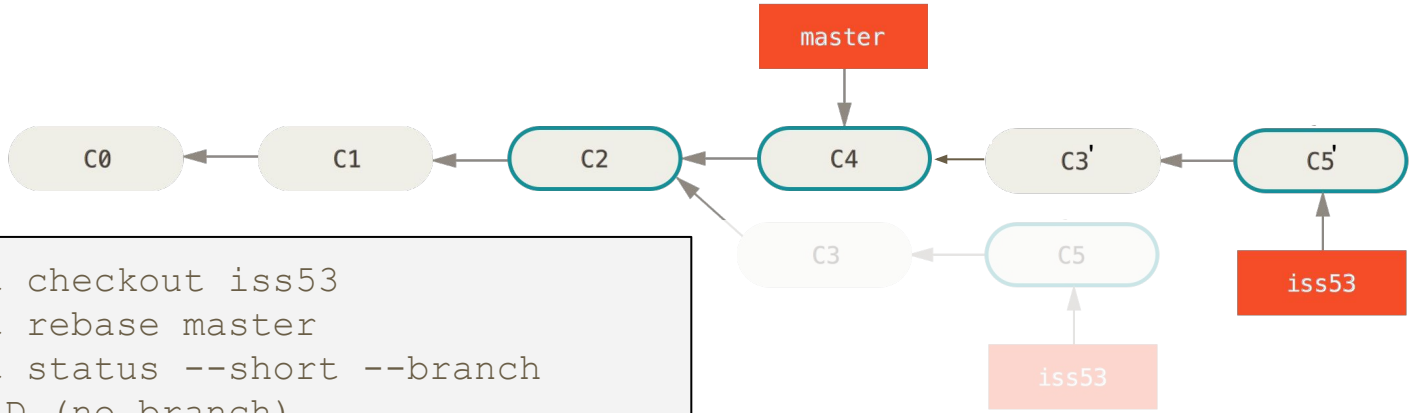
TP : git branch (rebase)




```
~$ git checkout iss53
~$ git rebase master
~$ git status --short --branch
## HEAD (no branch)
UU file.txt
?? file.txt.orig
```

3. Commandes basiques de GIT

TP : git branch (rebase)

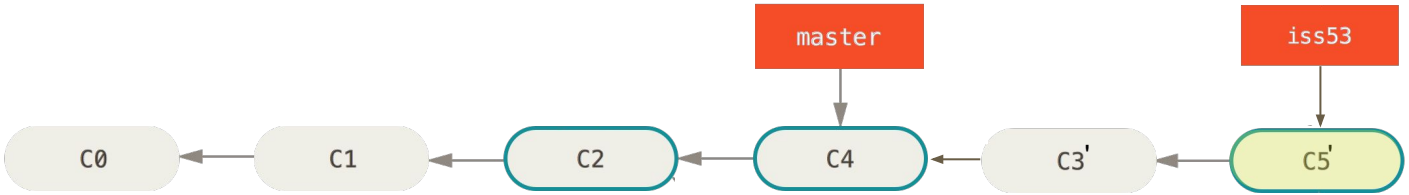



```
~$ git checkout iss53
~$ git rebase master
~$ git status --short --branch
## HEAD (no branch)
UU file.txt
?? file.txt.orig
```

```
~$ git branch -v
* (no branch, rebasing iss53) 9d873aa file-2 added
```

3. Commandes basiques de GIT

TP : git branch (rebase)

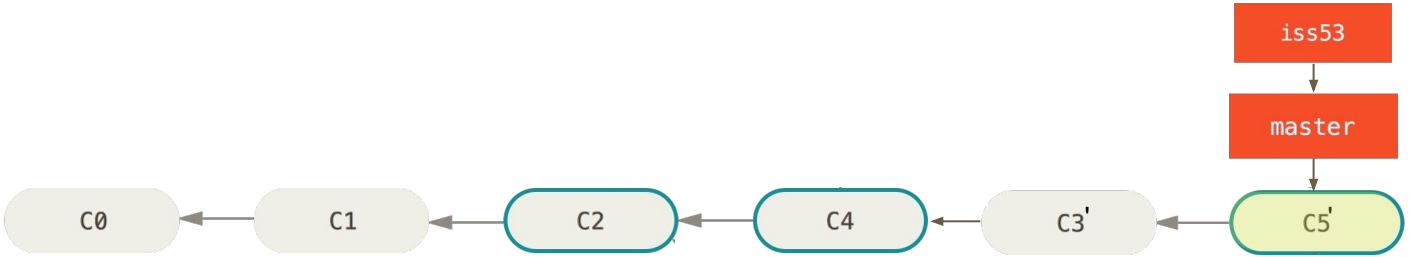



```
~$ git mergetool
...

~$ git rebase --continue


~$ git rebase --skip
~$ git rebase --abort
```

```
~$ git status --short --graph
* 277fe914 - (HEAD, iss53) mail > titi
* 9d873aa4 - file-2 added
* 86a5a411 - (master, hotfix) mail > bar
* 4a47382d - (tag: v1.0) remove f2
* bd53f07e - email here
* 7801bc71 - first commit
```

3. Commandes basiques de GIT

TP : git branch (rebase)

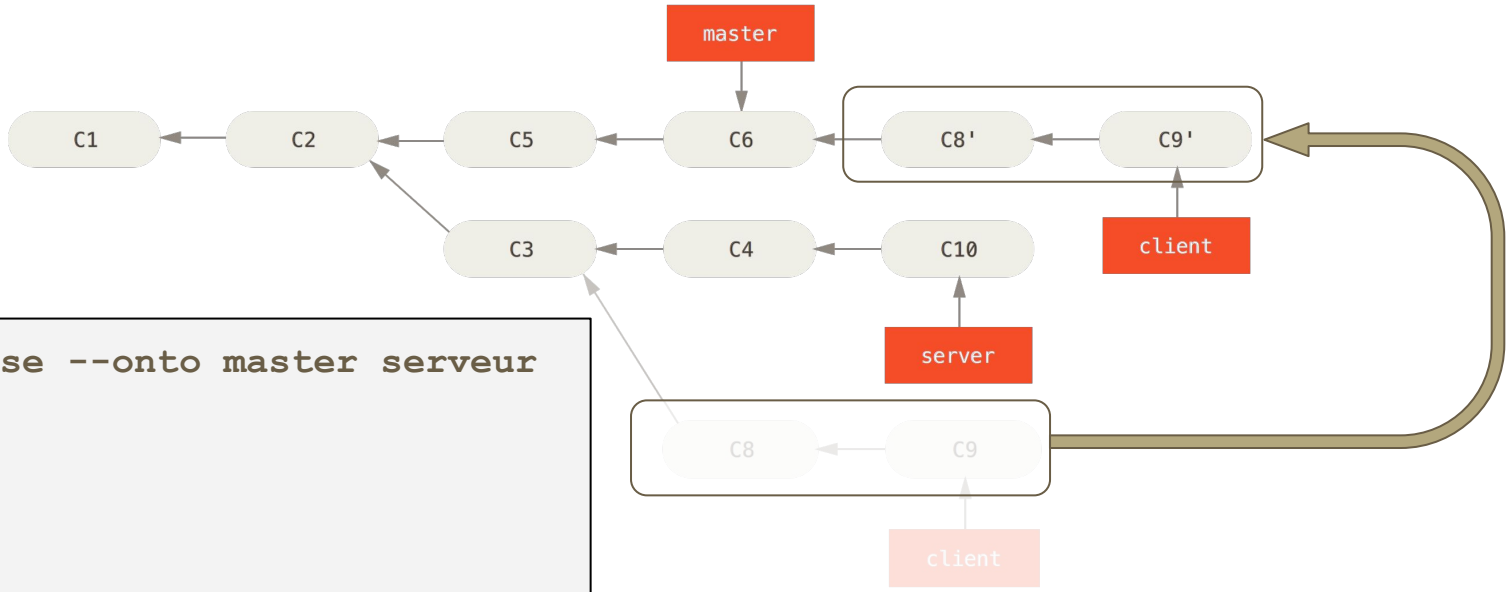



```
~$ git checkout master
~$ git merge iss53
Updating 86a5a41..277fe91
Fast-forward
 file-2.txt | 0
 file.txt   | 2 +-
 2 files changed, 1 insertion(+), 1
deletion(-)
 create mode 100644 file-2.txt
```

```
~$ git status --short --graph
* 277fe914 - (HEAD, master, iss53)
* 9d873aa4 - file-2 added
* 86a5a411 - (hotfix) mail > bar
* 4a47382d - (tag: v1.0) remove f2
* bd53f07e - email here
* 7801bc71 - first commit
```

3. Commandes basiques de GIT

TP : git branch (rebase)




```
~$ git rebase --onto master serveur
client
```

3. Commandes basiques de GIT

TP : git branch (rebase)

```
~$ git rebase ...
```

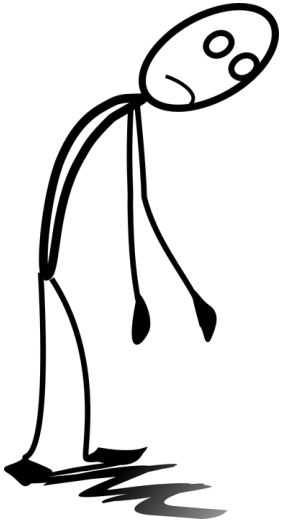
"Ne rebasez jamais des commits qui ont déjà été poussés sur un dépôt public.

Si vous suivez ce conseil, tout ira bien.

3. Commandes basiques de GIT

TP : git branch (rebase)

```
~$ git rebase ...
```

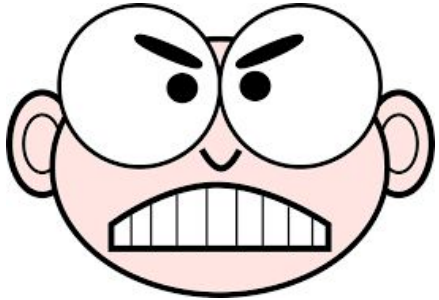


"Ne rebasez jamais des commits qui ont déjà été poussés sur un dépôt public.

Si vous suivez ce conseil, tout ira bien.

***Sinon**, de nombreuses personnes vont vous haïr et **vous serez méprisés par vos amis et votre famille**."*

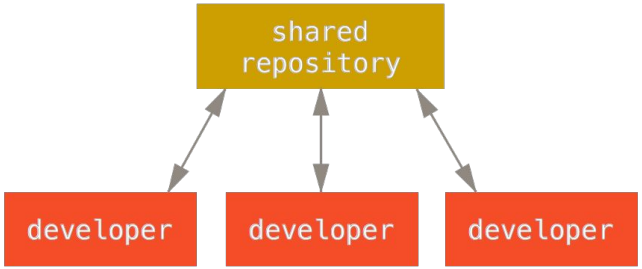
~=> git ci --amend



3. Commandes basiques de GIT

TP : git remote add

```
~$ cd git_wc
~$ git remote add origin \
   https://github.com/syjust/sjtp.mc-remote

~$ cat .git/config
...
[remote "origin"]
  url = https://github.com/syjust/sjtp.mc-
remote
  fetch = +refs/heads/*:refs/remotes/origin/*
...

~$ git fetch origin && git branch --all -vv
  hotfix
  iss53
* master
  remotes/origin/master
```



3. Commandes basiques de GIT

TP : git remote add

```
~$ git fetch origin jane john syl
  ...
  remotes/origin/jane
  remotes/origin/john
  remotes/origin/syl

~$ git checkout john
Branch john set up to track remote branch john
from origin

~$ git branch --all -vv
* john                    ad856f5 [origin/ john]
  master                  ad29e8c
  syl                     c85737a [origin/syl]
  remotes/origin/ john    ad856f5
  remotes/origin/ master  aa017d9
```

```
~$ cat .git/config
...
[branch "john"]
  remote = origin
  merge = refs/heads/john
...
```

3. Commandes basiques de GIT

TP : git remote add

```
~$ git checkout origin/master
You are in 'detached HEAD' state...

~$ git branch --all -vv
* (detached from origin/master) aa017d9
  john                    ad856f5 [origin/john]
  master                  ad29e8c
  syl                     c85737a [origin/syl]
  remotes/origin/john     ad856f5
  remotes/origin/master   aa017d9

~$ git checkout -b git-master
~$ git branch --set-upstream-to=origin/master \
   git-master
```

```
~$ cat .git/config
...
[branch "git-master"]
  remote = origin
  merge = refs/heads/master
...
```

3. Commandes basiques de GIT

TP : git remote add

```
~$ git checkout origin/master
You are in 'detached HEAD' state...

~$ git branch --all -vv
* (detached from origin/master) aa017d9
  john                          ad856f5 [origin/john]
  master                        ad29e8c
  syl                           c85737a [origin/syl]
  remotes/origin/john           ad856f5
  remotes/origin/master         aa017d9

~$ git checkout -b git-master
~$ git branch --set-upstream-to=origin/master \
   git-master

~$ git branch git-master origin/master
```

```
~$ cat .git/config
...
[branch "git-master"]
  remote = origin
  merge = refs/heads/master
...
```

3. Commandes basiques de GIT

TP : git remote ssh

```
~$ [ ! -d ~/.ssh ] && mkdir ~/.ssh
~$ cp security/github/id_rsa* ~/.ssh
~$ cat security/github/ ssh-config >> ~/.ssh/config
```

```
~$ ssh-keygen -p -f ~/.ssh/id_rsa-sjtp.mc-2016
Enter old passphrase: mc-2016
Key has comment '~/.ssh/id_rsa-sjtp.mc-2016'
Enter new passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved with the new passphrase.
```

3. Commandes basiques de GIT

TP : git remote clone

```
~$ cd ../
~$ git clone \
   ssh://github.com/syjust/sjtp.mc-remote

~$ cd sjtp.mc-remote && git branch --all -vv

* master                 aa017d9 [origin/master]
  remotes/origin/HEAD    -> origin/master
  remotes/origin/jane    97b2ef5
  remotes/origin/john    ad856f5
  remotes/origin/master aa017d9
  remotes/origin/syl     c85737a
```

git pull origin branchname =

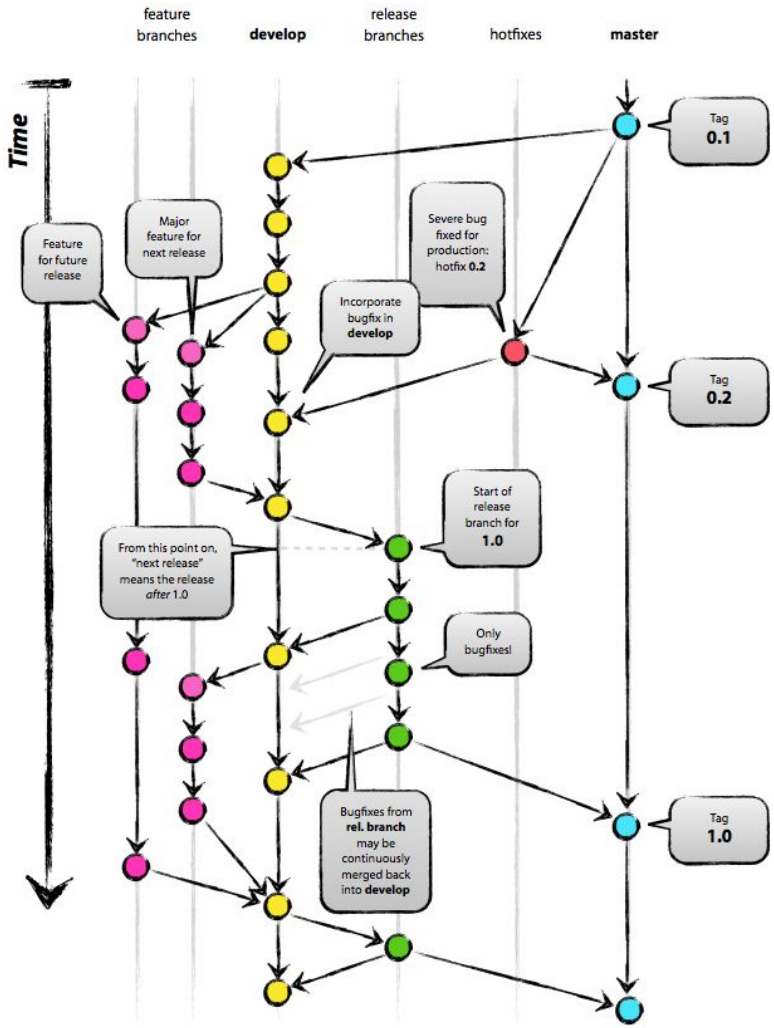
git fetch origin branchname +
git merge origin/branchname

4. Bonnes pratiques et conventions GIT

- Branche par défaut : **master**
- Dépôt (**remote**) par défaut : **origin**
- Pointeur du système de fichier : **HEAD** (notion de tree)
- Plusieurs **commit** par jour (le message de 60 *char* doit suffire)
  - 1 titre de 60 char
  - des commentaires sur 80 char en sautant une ligne après le titre
- **1 push** par jour
- les commit pushés sur le dépôt distant doivent être propres

4. Bonnes pratiques...

- Une **branch** par feature
- Ne jamais commiter dans **master**
- Une branche de **dev** (très) active
- Chaque **merge** dans master => **tag**



4. Bonnes pratiques…

Contribution : pull request

- GIT
  - ! lock / ! unlock
  - branch = rwx