2. Migrer de SVN vers GIT

Utilisez GIT en multi-dépôt et modifiez votre historique

Jour 1 / Après-midi

1. Terminologie GIT vs SVN (QR)
2. Migrer de SVN vers GIT (Retour d’expérience)
3. La commande git-svn (TP)
4. Gestion de plusieurs dépôts distants (Remote)
5. QR : commandes basiques de GIT / migration de SVN > GIT

1. Terminologie GIT vs SVN

Questions / Réponses

- copie de travail (dépôt local) → copie de travail
- dépôt (distant ou local) → dépôt (distant)
- add / mv / rm (local only) → add / mv / rm (can be remote)
- commit (local) → commit (distant)

1. Terminologie GIT vs SVN

Questions / Réponses

- copie de travail (dépôt local)
- dépôt (distant ou local)
- add / mv / rm (local only)
- commit (local)

- copie de travail
- dépôt (distant)
- add / mv / rm (can be remote)
- commit (distant)

QUIZZ cmd

 => 

OR

 <= 

1. QUIZZ  => 

Questions / Réponses

pull	
clone	
checkout (local)	
remote	
archive	
branch \ tag	
(chmod)	lock / unlock
gitattributes	
.git/	
(branch)	cp

1. QUIZZ  <= 

Questions / Réponses

	update
	checkout (distant)
	revert (distant)
	switch
	import
	cp branches/ \ .cp/tag/version-0.0.1
	lock / unlock
	prop{set,del,edit,list}
	.svn/
	cp

1. Terminologie GIT vs SVN

Questions / Réponses

pull	update
clone	checkout (distant)
checkout (local)	revert (distant)
remote	switch
archive	import
branch \ tag	cp branches/ \ .cp/tag/version-0.0.1
(chmod)	lock / unlock
gitattributes	prop{set,del,edit,list}
.git/	.svn/
(branch)	cp

2. Migrer de SVN vers GIT

- Retour de *(votre)* expérience
- Les différentes méthodes

2. Migrer de SVN vers GIT

Retour d'expérience

- Lenteur (limiter la taille du dépôt local)
- Garder l'historique
- Garder SVN à jour ou migrer complètement sur GIT

2. Migrer de SVN vers GIT

Les différentes méthodes

1. réécrire les delta SVN dans GIT (solution employée pour une migration vers HG)

```
r=`svn info | awk -F: '$1 ~ /Revision/ {print $2}'`

for rev in `seq 1 $r` ; do
    mess=`svn log -r $rev`
    svn export -r $rev \
        && git add * \
        && git ci -m "$mess"
done
```

2. Migrer de SVN vers GIT

Les différentes méthodes

2. Utiliser le multi-Dépôt et la commande git-svn

```
git-svn
```

1 migrer par étape (équipe par équipe) et **garder SVN & GIT** ensemble pendant un temp

2 migrer tout et ne **plus utiliser SVN**

3. La commande git-svn

Options de la commande git-svn

- clone = *svn checkout*
- fetch + rebase = *svn update*
- dcommit = *svn commit*
- clone -T -t -b *vs* -s (--stdlayout) = *svn /trunck /tags /branches*

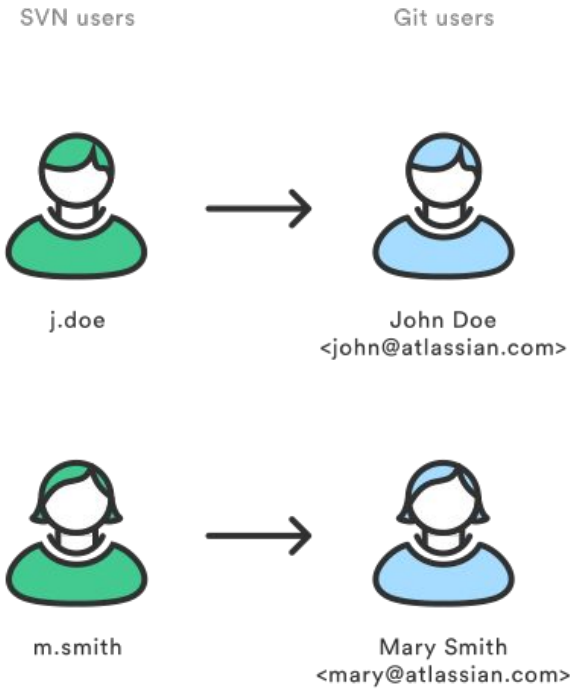
3. Migrer

TP : git-svn

- Préparer (options & métadonnés des commits)
- Convertir (dépôt SVN > dépôt local GIT)
- Synchroniser
- Partager
- Migrer

3. Migrer 1 & 2

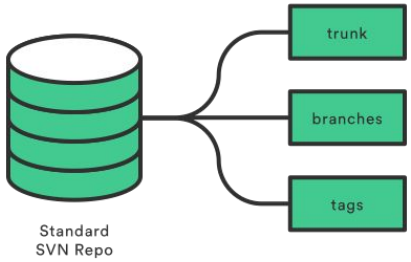
TP : git-svn > Préparer

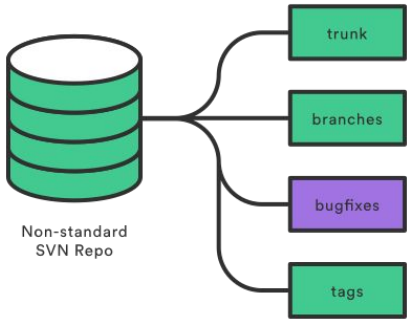



```
~$ svn co \
   --username=syjust \
   --password=sjtp-MC-2016 \
   http://svn.assembla.com/svn/sjtp.mc \
   svn_wc

~$ cd svn_wc

~$ svn log ^/ --xml \
   | grep -E "^<author" \
   | sed 's_<author>\([^<]*\)</author>_\1' \
   | sort -u \
   > users.txt

~$ sjtp.mc-init/bin/migration/extract-authors.sh
```

3. Migrer 1 & 2

TP : git-svn > convertir I

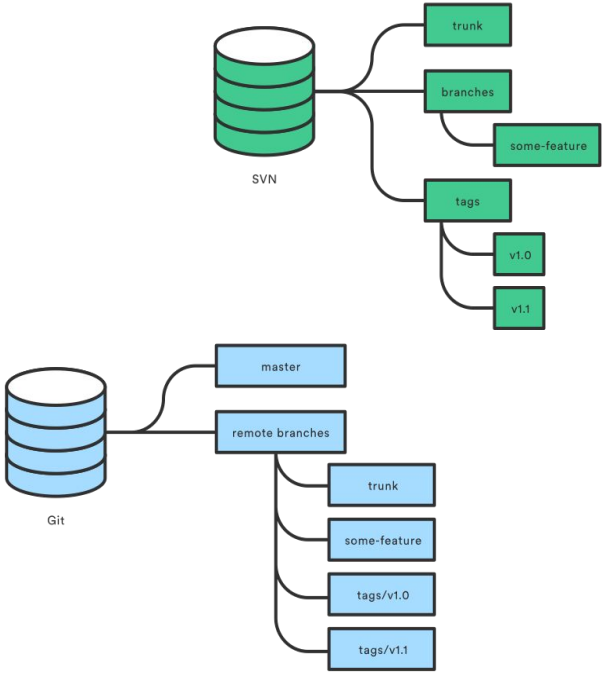





```
~$ git svn clone --stdlayout

~$ git svn clone \
   -T /trunk
   -t /tags
   -b /branches
   -b /bugFix
```

3. Migrer 1 & 2

TP : git-svn > convertir II




```
~$ git svn clone --stdlayout \
   --prefix=git-svn/ \
   --authors-file=users.txt \
   http://svn.assembla.com/svn/sjtp.mc \
   git_svn_wc

~$ cd git_svn_wc

~$ git br --all
    * master
      remotes/git-svn/dev
      remotes/git-svn/prod
      remotes/git-svn/staging
      remotes/git-svn/tags/v-0.0.1
      remotes/git-svn/trunk
```

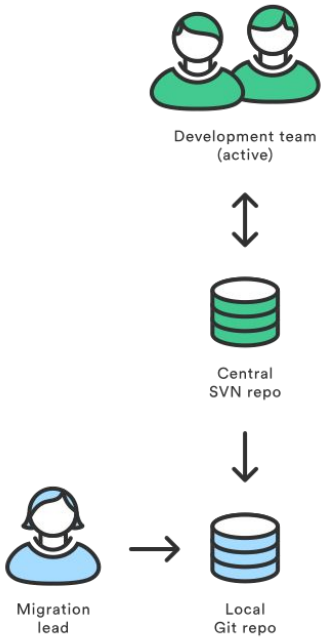
3. Migrer 1 & 2

TP : git-svn > convertir III

```
~$ git log
    commit d4af13458f4003c56b60fa385534ce41aa4bb232
    Author: syjust <syjust@maincare.com>
    Date:   Fri Apr 15 10:24:04 2016 +0000

        config.txt in trunk


git-svn-id: http://svn.assembla.com/svn/sjtp.mc/trunk@5 46fb47e-f0d5-4810-b810-7b15cde51753
...

~$ git svn clone --no-metadata ... git_svn_wc
```

3. Migrer 1

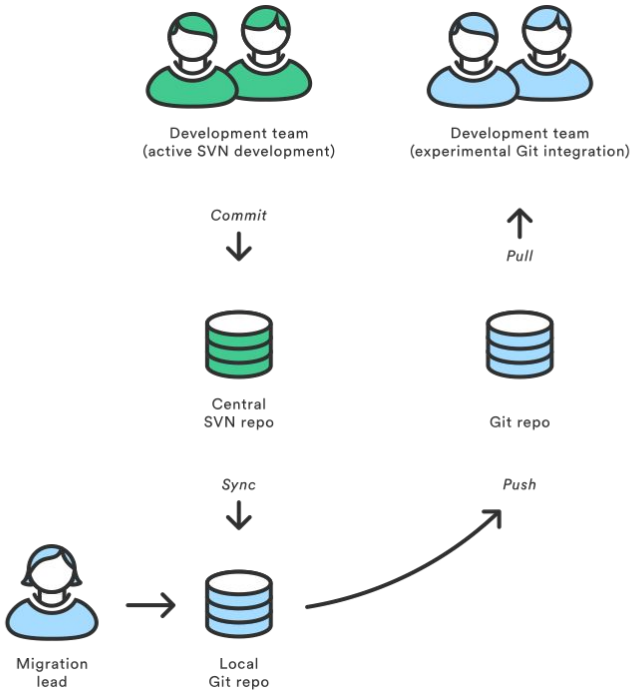
TP : git-svn > synchroniser




```
~$ git branch trunk git-svn/trunk
[/!\ git branch -avv track is not visible]

[faire une modif et commiter dans svn_wc]

~$ git svn fetch git-svn
   M       afile.txt
   r12 = 0405c15e (refs/remotes/git-
svn/trunk)

~$ git checkout trunk
~$ git rebase git-svn/trunk
   First, rewinding head to replay your work
   on top of it…
   Fast-forwarded trunk to git-svn/trunk.
```

3. Migrer 1

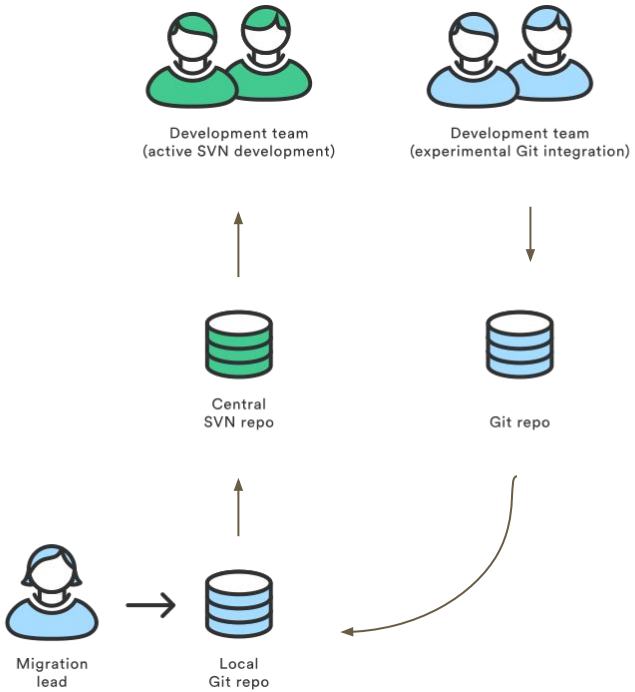
TP : git-svn > partager




```
~$ git init --bare ../empty.git
~$ git remote add origin ../empty.git

[faire des liens pertinents entre les
branches git et svn]

~$ git push -u origin master \
   [branch...] --tags

~$ git svn fetch git-svn && git co trunk
~$ git rebase git-svn/trunk

~$ git checkout master
~$ git merge trunk
~$ git push origin master
```

3. Migrer 1

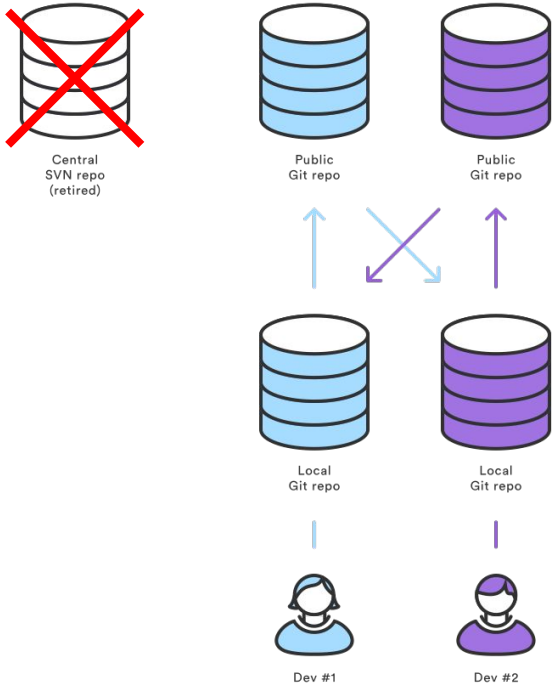
TP : git-svn > partager




```
~$ git merge trunk master
~$ git push origin master

~$ git pull

~$ git checkout trunk
~$ git merge master
~$ git svn dcommit
```

3. Migrer 1 & 2

TP : git-svn > migrer




```
~$ branches=`git for-each-ref...`
~$ tags=`git for-each-ref...`

[seulement au moment de migrer
 transformer les branches virtuelles en
 tag et branches git]

~$ foreach b in $branches ; do
~$   git branch $b git-svn/$b
~$   git branch -d git-svn/$b
~$ done

~$ ./bin/migration/sanitize-branches.sh
~$ ./bin/migration/sanitize-tags.sh
```

3. Migrer

Limiter l'historique

```
[Avant de synchroniser le dépôt]
~$ git clone URL --branch branch_name --single-branch [folder]
~$ git clone --depth depth remote-url

[directement au niveau du dépôt]
~$ git filter-branch --tree-filter 'rm -rf /path/to/asset/folder' HEAD
```

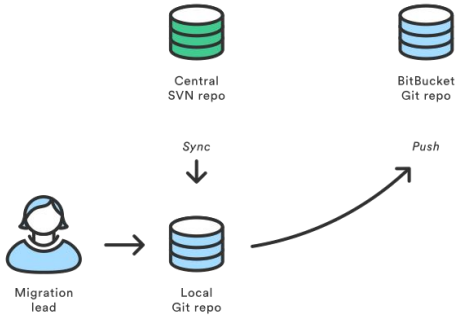
3. La commande git-svn

Avantages et inconvénients

1

migrer par étape (équipe par équipe) et **garder SVN & GIT** ensemble pendant un temp

-- travail d'intégration fastidieux
-- connaître à fond **git-svn**
++ transparence d'utilisation pour les développeurs



2

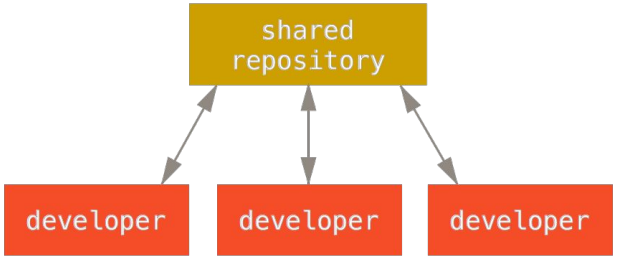
migrer tout et ne **plus utiliser SVN**

-- tous les développeurs doivent connaître à fond **git**
++ un seul système est toujours plus facile à gérer

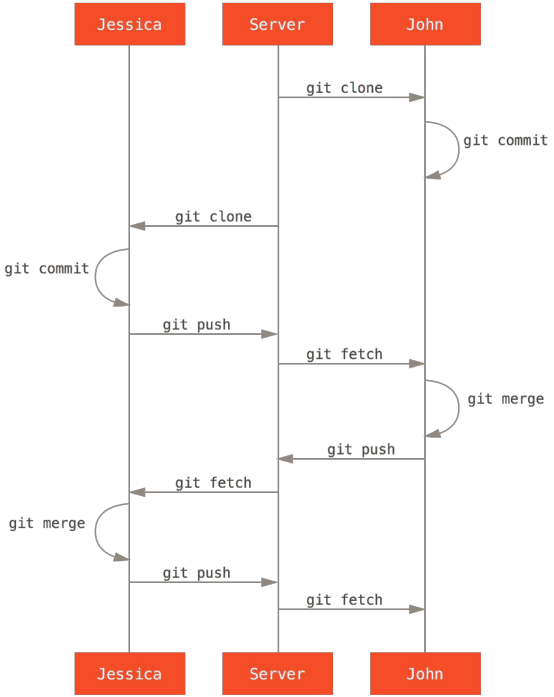
4. Gestion multi-repos

private : multi vs single

Contribuer sur un seul dépôt



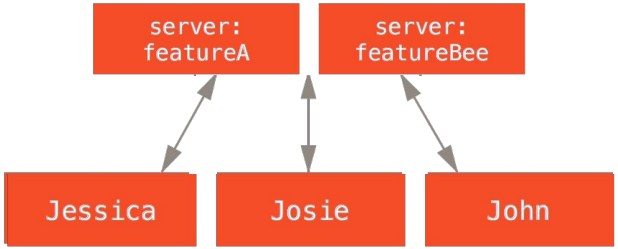
1. Petite équipe
2. Projet privé (accès en écriture au dépôt)



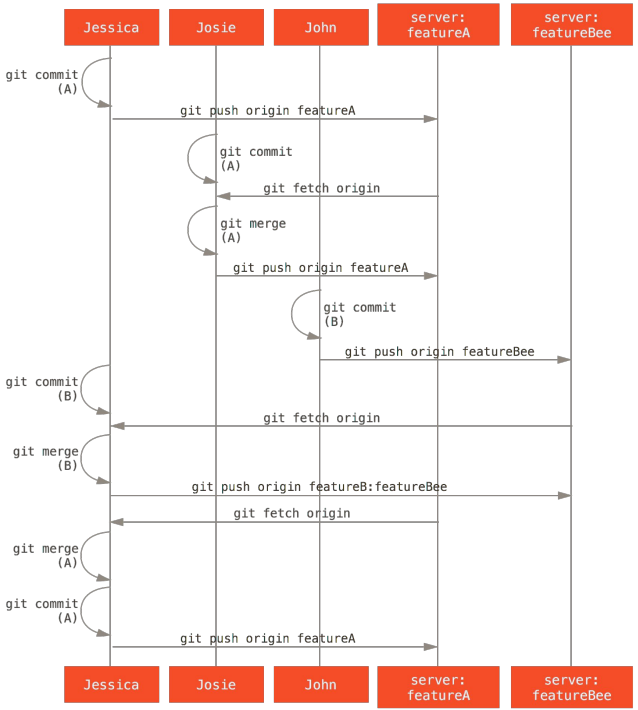
4. Gestion multi-repos

private : multi vs single

Contribuer sur plusieurs dépôt

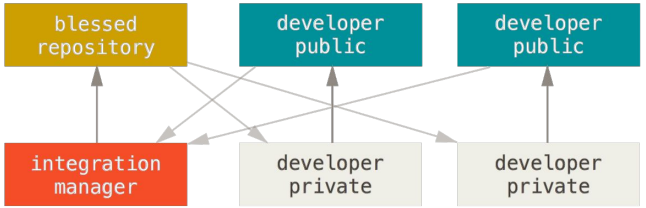


1. Équipes plus importantes (sous-équipe)
2. Chaque équipe travaille sur une feature
3. Il y'a un gestionnaire d'intégration
4. Toujours en Projet privé



4. Gestion multi-repos

public : integration manager



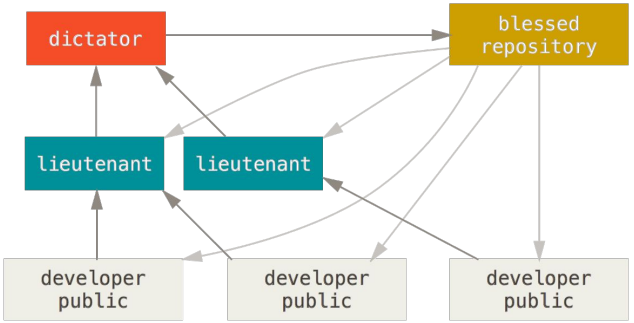
Mode du gestionnaire d'intégration (*gitHub*)

1. Le mainteneur du projet pousse vers son dépôt public.
2. Un contributeur clone ce dépôt et introduit des modifications.
3. Le contributeur pousse son travail sur son dépôt public.
4. Le contributeur envoie au mainteneur un e-mail de demande pour tirer ses modifications depuis son dépôt. (**pull-request)**
5. Le mainteneur ajoute le dépôt du contributeur comme dépôt distant et fusionne les modifications localement.
6. Le mainteneur pousse les modifications fusionnées sur le dépôt principal.

4. Gestion multi-repos

public : dictator

Mode dictateur et ses lieutenants (*Linux*)



1. Les simples développeurs travaillent sur la branche thématique et rebasent leur travail sur master. La branche master est celle du dictateur.
2. Les lieutenants fusionnent les branches thématiques des développeurs dans leur propre branche master.
3. Le dictateur fusionne les branches master de ses lieutenants dans sa propre branche master.
4. Le dictateur pousse sa branche master sur le dépôt de référence pour que les développeurs se rebasent dessus.

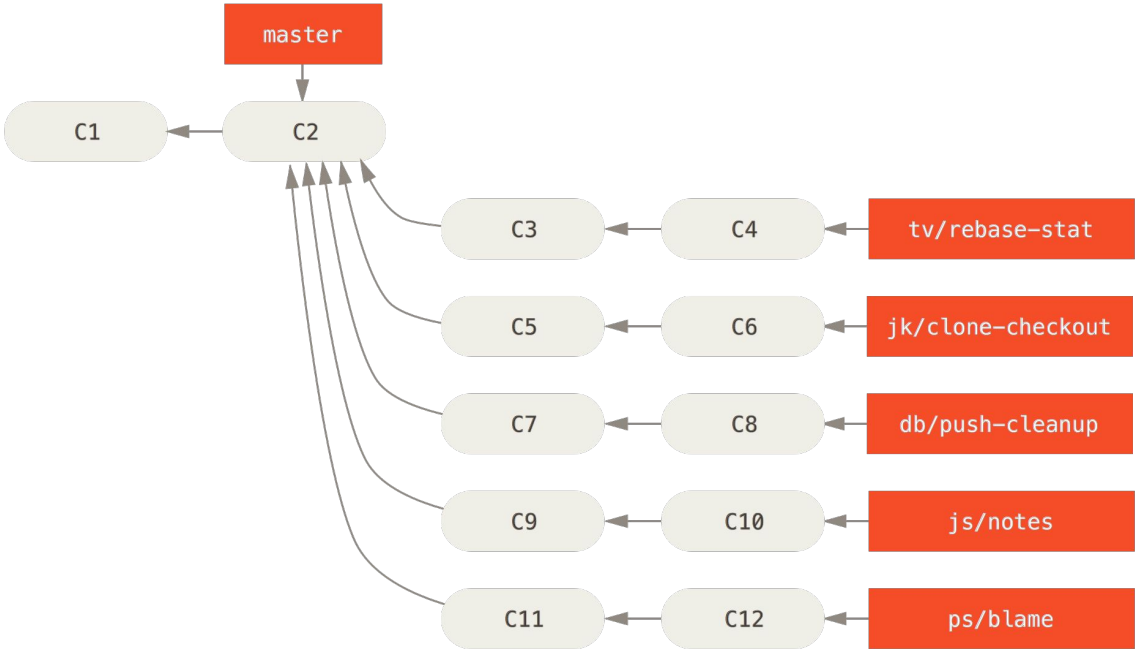
4. Gestion multi-repos

maintenance

Exemple du projet **GIT**

(4 branches au long court)

- master
- next
- pu (proposed updates)
- maint (backport)
- **+ branches thématiques**



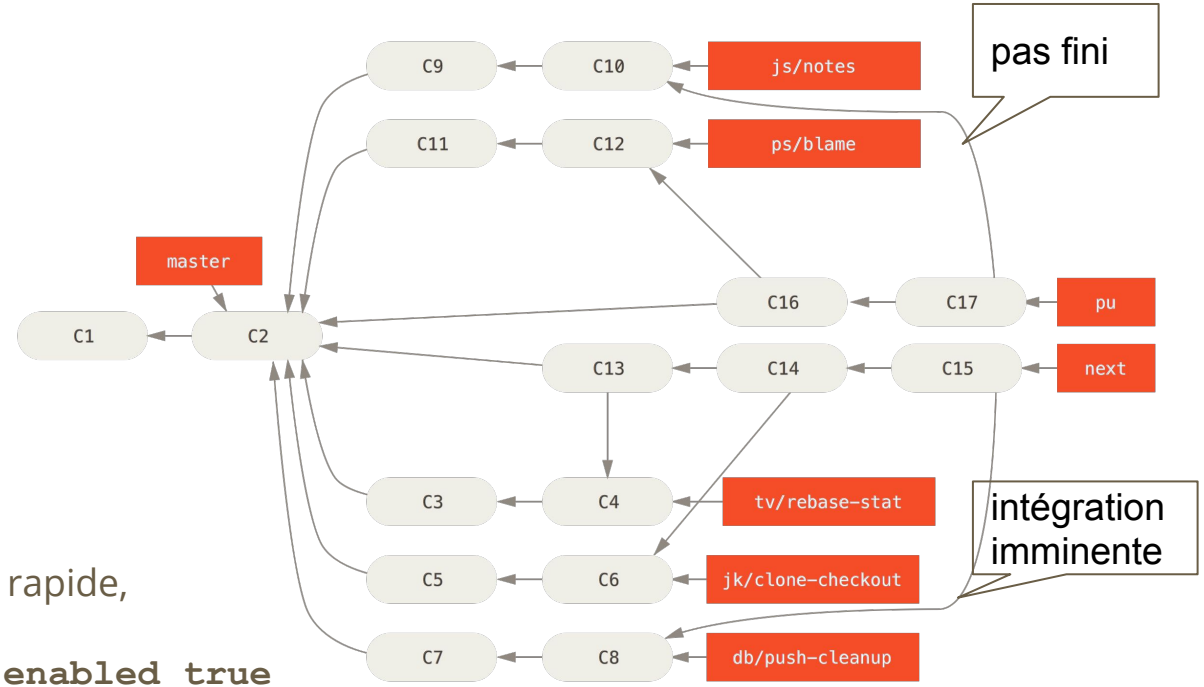
4. Gestion multi-repos

maintenance

Exemple du projet **GIT**

(4 branches au long court)

- **master**
- **next**
- **pu** (proposed updates)
- maint (backport)
- + branches thématiques



pu est rebasé très souvent
next est rebasé assez souvent,
master évolue toujours en avance rapide,

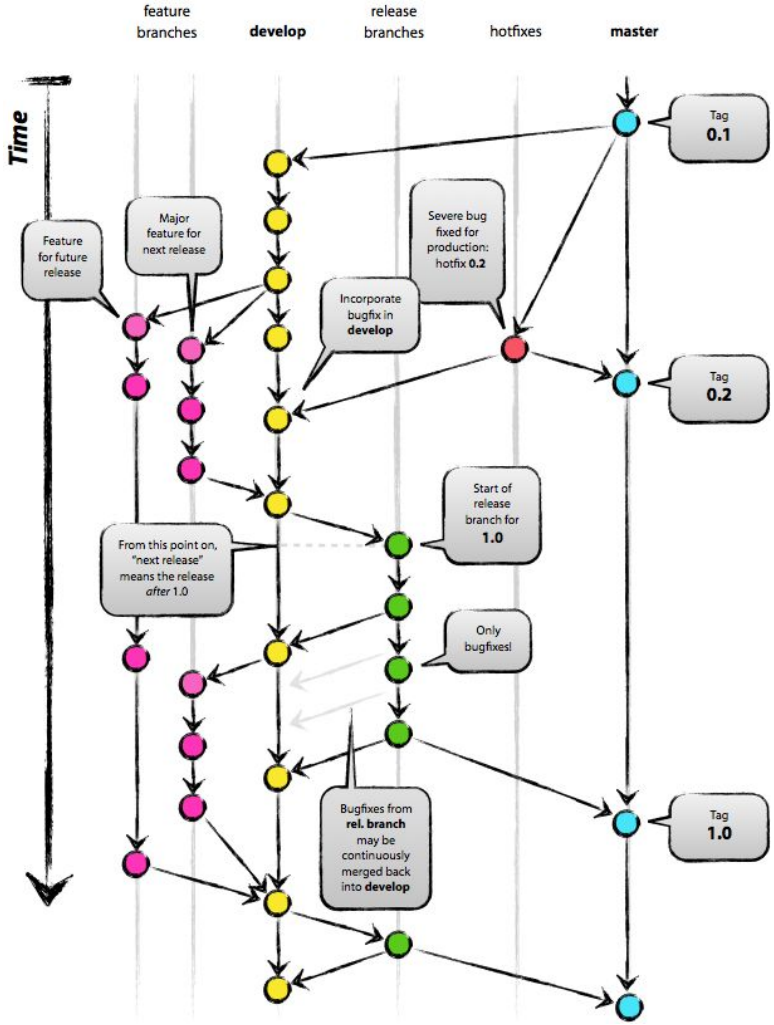
```
git config --global rerere.enabled true
```

4. Gestion multi-repos

Objectif : historique propre et linéaire

On peut décider que l'historique montre **ce qui s'est réellement passé**.

Ou décider que l'historique montre **l'évolution du code qu'on aurait souhaité**.



5. Questions / Réponses

- commandes basiques GIT
- migration SVN > GIT
- les dépôts distants et la maintenance des dépôts
- ...