Git : la gestion de versions décentralisée

Association GUILDE

http://www.guilde.asso.fr

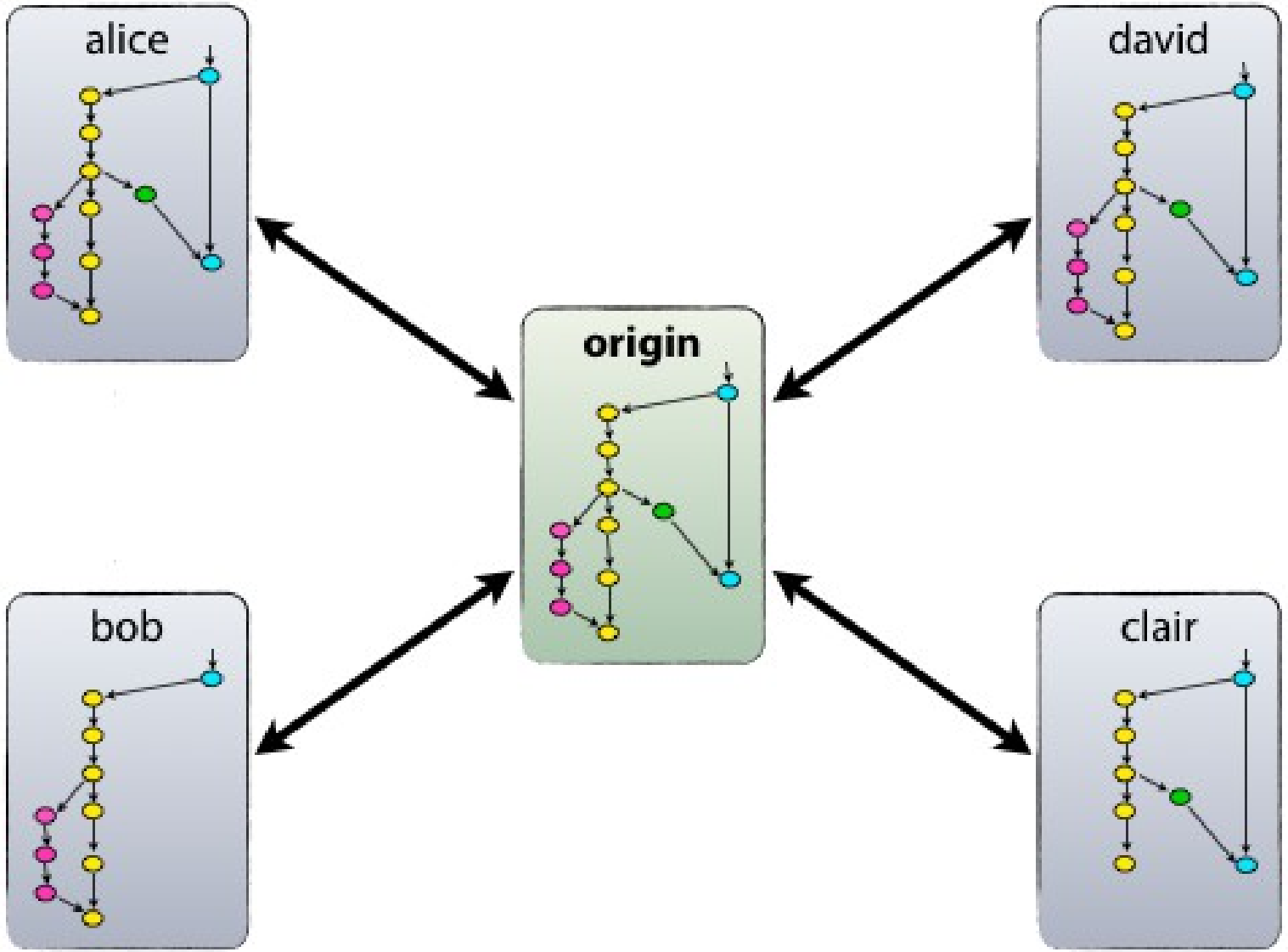
La gestion de versions

Extrait de la page Wikipédia :

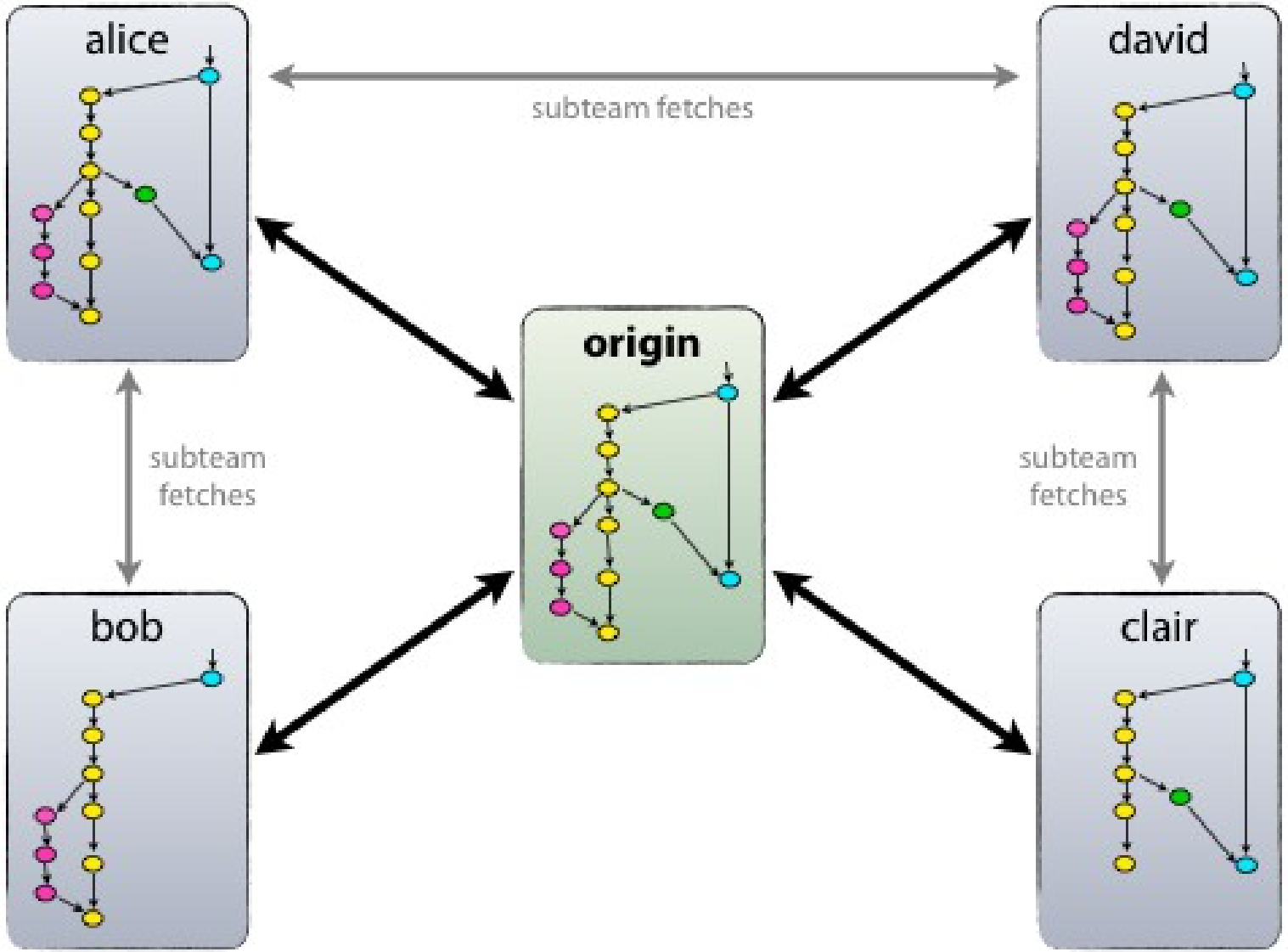
« *La gestion de versions (en anglais version control ou revision control) consiste à maintenir l'ensemble des versions d'un ou plusieurs fichiers.* »

Versions du Projet	Version fichier A	Versions fichier B	Versions fichier C
Toto 1.0	1.0	1.0	1.0
Toto 1.1	1.1		1.1

Gestion de versions centralisée



Gestion de versions décentralisée



Principes généraux de Git

Historique - dernière version du dépôt (*commit*)

Fichier A – version 1.1
Fichier B – version 1.0
Fichier C – version 1.1

Index – prochain *commit*

Fichier A – version 1.1
Fichier B – version 1.1
Fichier C – version 1.1

Répertoire de travail – fichiers visibles

.git/
Fichier A – version 1.2
Fichier B – version 1.1
Fichier C – version 1.1

Commandes : Obtenir de l'aide

```
$ man git

$ man git-<commande>
```

```
$ git help

$ git help <commande>
```

Commandes : État actuel

```
$ git status
```

```
fernando@machina:/tmp/toto$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   Fichier_B
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   Fichier_A
#
fernando@machina:/tmp/toto$
```

Commandes : Configuration

```
$ git config --global user.email fernando@demo-tic.org
$ git config --global color.ui auto
```

~/.gitconfig

```
$ git config user.email fernando@no-log.org
```

.git/config

```
$ git config user.email
```

Commandes : initialisation du dépôt

```
$ git init .

$ git init --bare toto.git
```

```
$ git clone ssh://rosette/home/fernando/toto.git

$ git clone file:///home/fernando/toto.git titi
```

```
$ git clone git://gitorious.org/myplan
```

Commandes : manipuler l'index

```
$ git add Fichier_C

$ git stage Fichier_A

$ git add .


$ git rm --cached Fichier_C

$ git reset HEAD Fichier_C

$ git rm Fichier_C


$ git status ##rappel
```

Commandes : commit

```
$ git commit

$ git commit -a

$ git commit -m "Commit message."


$ git commit Fichier_C


$ git commit --amend
```

Commandes : voir les différences

```
$ git diff
```

```
$ git diff HEAD
$ git diff <commit id>
```

```
$ git diff --cached
```

```
$ git diff -- Fichier_C
```

Commandes : voir l'historique

```
$ git log
```

```
$ git log -- Fichier_C
```

```
$ git log --graph --oneline --decorate
```

```
$ gitg
```

Commandes : voir les différences

```
$ git diff
```

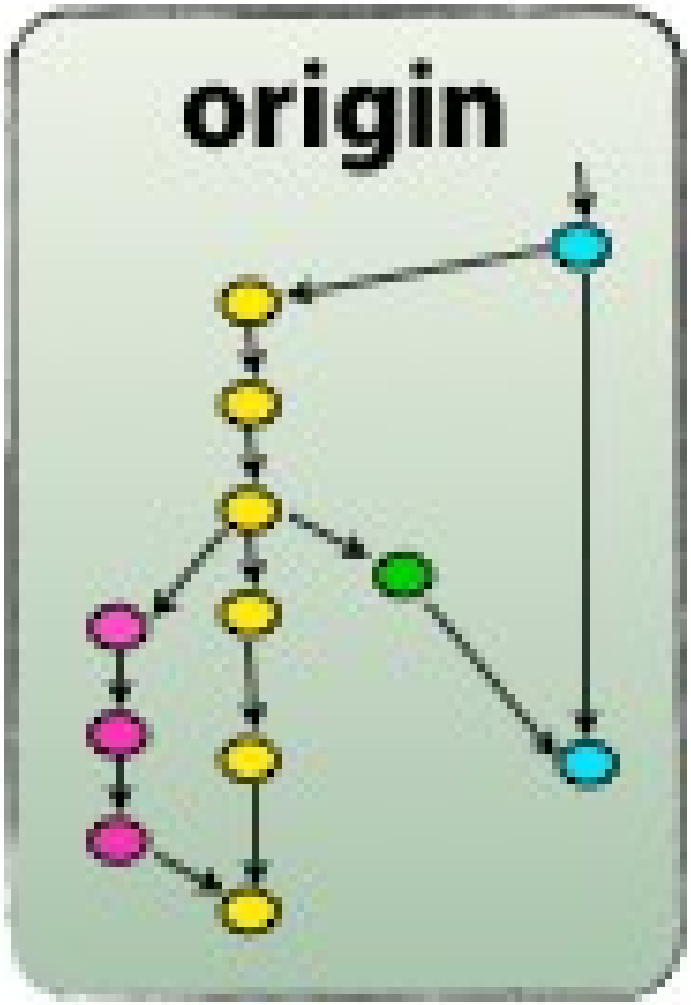
```
$ git diff HEAD
$ git diff <commit id>
```

```
$ git diff --cached
```

```
$ git diff -- Fichier_C
```

Commandes : branches

```
$ git branch -a

$ git branch new_fix

$ git checkout new_fix


$ git branch -m fix

$ git branch -d fix


$ git checkout -b hot_fix master
```



Commandes : fusions

```
$ git rebase -i # branche hot_fix
```

```
$ git merge --no-ff hot_fix # br. master
$ git rebase master
```

```
$ git cherry-pick <commit>
```

Commandes : dépôts distants

```
$ git clone <url>

$ git remote

$ git remote add <nom> <url>


$ git fetch --all

$ git pull


$ git push <remote> <branche> --dry-run
```

Commandes : bissection

```
$ git bisect start

$ git bisect good <bon_commit>

$ git bisect bad <mauvais_commit>


$ git bisect start <mauvais> <bon>

$ git bisect run <commande_de_test>
```

Bonnes pratiques

- Tester avant de mettre à jour le référentiel
- User (et abuser) des branches
- Faire des petits *commit*, utiliser `git rebase -i`
- Mettez au point des règles de fonctionnement communes

L'écosystème de git

gitg

gitk

gitX

Git Extensions

Git SVN

…

Gource

Git Flow

http://www.gitorious.org

http://github.com

Questions ?

Et vous, comment utilisez-vous git ?

Remerciements

Vincent Driessen, diagramme diapo 4 et 15 CC-BY-SA