

Des workflows, pour quoi faire ?



Antoine Pérus - LAL

1er octobre 2015



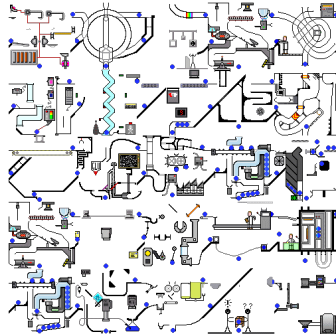
AP ?

```
var me = new Speaker() {  
  Fullname = "Antoine Pérus",  
  Laboratory = "LAL - Orsay",  
  Email = "perus AT lal.in23.fr",  
  Title = "HEP (old) developer"  
}
```



Préambule

Ceci n'est pas une présentation à propos de la tuyauterie
Git ...



Préambule

Ce n'est pas non plus une introduction



Bazaar

ni un comparatif de leurs mérites respectifs ...



Workflows

- Kezako ?
- Heu ...



Workflows

- Anglicisme pour *flux de travaux*
- Représentation d'une suite de tâches ou opérations effectuées par une personne, un groupe de personnes, un organisme, etc.

Le terme **flow** (flux) renvoie au passage du produit, du document, de l'information ... d'une étape à l'autre.

Aussi bien :

- tout type de processus de travail et de publication
- la description et l'ordonnancement des tâches au sein d'un collectif

Workflows

- Permettent d'acquérir une culture et une pratique commune
- Garantissent une cohérence dans le travail, en équipe ou seul
- Peuvent être propres à une personne, à un groupe quel que soit sa taille
- Une fois définis, ils peuvent être implémentés et intégrés pour en faciliter leur utilisation

Les branches

- Simple divergence de code
- Permettent d'avoir plusieurs lignes de développement parallèles
- On peut éventuellement nommer les branches
- Les branches peuvent fusionner (*fusion/merge*)
- La fusion peut créer des conflits qu'il faut arbitrer
- Les branches sont ou ne sont pas permanentes et globales



Objectifs

- Organiser les processus de travail
 - Séparer les versions de production des versions de développement
 - Limiter les conflits, les compilations cassées
 - Permettre une gestion efficace des bugs
 - Organiser le code-review
 - Minimiser les problèmes entre développeurs

Comment ?

**En créant des branches spécifiques
pour chaque usage**

Workflows et contexte

- Un ensemble de tâches définit un contexte
- Chaque contexte est propre à une équipe qui met en oeuvre cet ensemble
- On définit/explicite un workflow pour chaque contexte
 - un workflow pour organiser le développement
 - un workflow pour la publication et le *code-review*
 - ...

Workflow et codage

Le workflow qui organise les tâches de développement, de codage va :

- s'appuyer sur l'outil de gestion de version utilisé
- être contraint par l'outil
- ... et dépendre bien sûr de tout un tas d'autres choix indépendants de l'outil

Workflow et publication

Le workflow qui organise la publication et le *code-review* va s'appuyer sur :

- l'outil de gestion de version utilisé
- et sur l'outil de publication, typiquement la forge sociale utilisée pour partager et exposer le projet

Workflows de développement



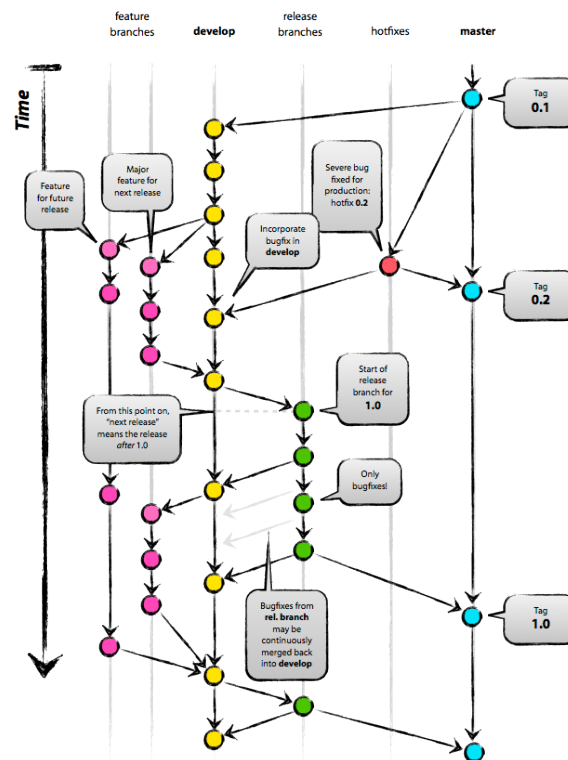
Git-flow

Proposé par [Vincent Driessen](#).

Un workflow *universel*, mais plutôt pour des projets dont les déploiements (*releases*) sont relativement espacés

- Deux branches 'historiques'
 - "master" / "default"
 - "develop"
- des branches de 'features'
- des branches de 'releases'
- des branches de maintenance ('hotfix')

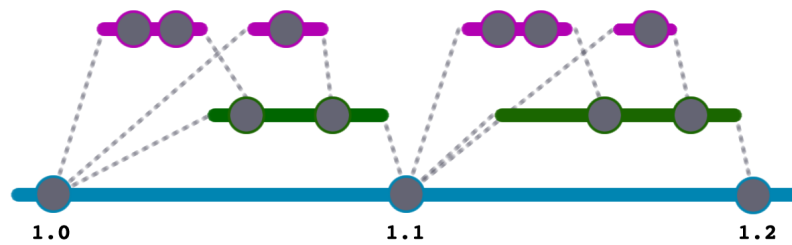
Git-flow



Branch-per-feature

Proposé par [Adam Dymitruk](#)

- une branche 'historique'
- des branches de 'features' synchronisées
- des branches d'intégration également synchronisées

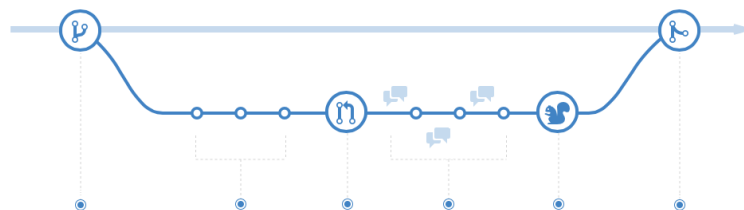


GitHub flow

Proposé par [Scott Shacon](#)

Un workflow léger pour des projets dont le déploiement est fait très régulièrement

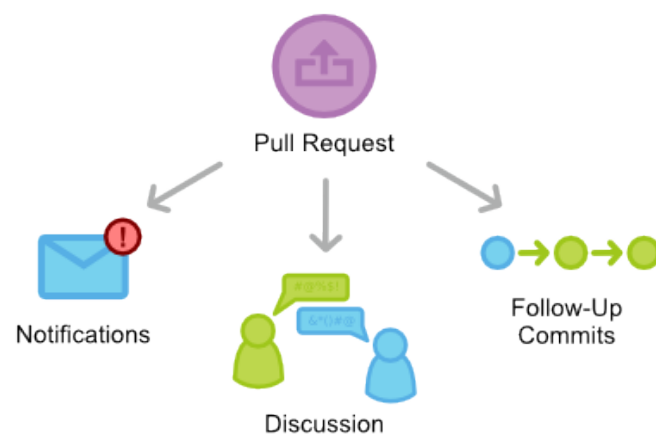
- une branche 'historique'
- des branches de 'features' avec *code-review* et déploiement



Workflow de publication



Pull-Request



Outils et intégration

L'utilisation d'outils simplifie nettement un processus complexe ...

- scripts
(gitflow.sh, extension hgflow, git_bpf, ...)
- intégration dans des interfaces graphiques
(SourceTree, ...)
- intégration dans des interfaces Web
(GitHub, GitLab, Bitbucket, RhodeCode, ...)

21

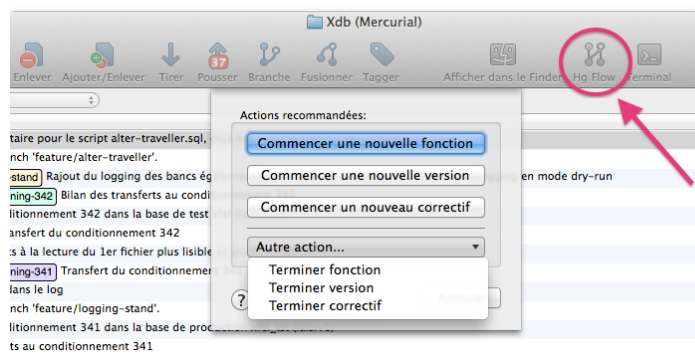
Script gitflow

```
> git flow feature
---- git flow feature
---- option
-v          -- Verbose (more) output
checkout    -- Checkout local feature branch.
diff        -- Show all changes.
finish      -- Finish a feature branch.
list        -- List all your feature branches. (Alias to `git flow featu
publish     -- Publish feature branch to remote.
pull        -- Pull changes from remote.
rebase      -- Rebase from integration branch.
start       -- Start a new feature branch.
track       -- Checkout remote feature branch.
```

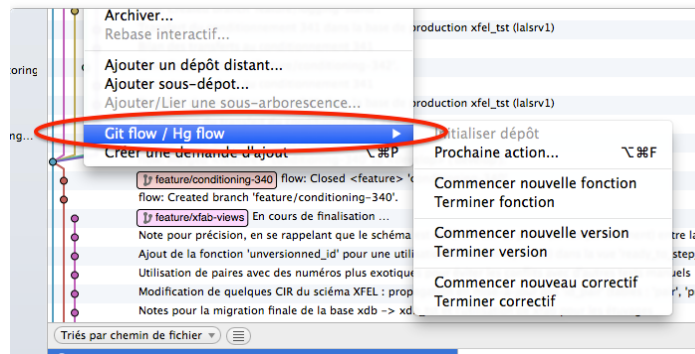
Extension hgflow

```
(venv) (develop)-->[+37] [?]> hg flow
flow: Currently open branches:
flow: <master> : default
flow: <develop>: develop*
flow: <feature>: feature/datafiles-multi-pass
                    feature/doc
                    feature/init-transfert-process-refactoring
                    feature/logging-stand
                    feature/test-get-coupler-conditioning-times
                    feature/xfab-views
```

Intégration SourceTree



Intégration SourceTree



Questions ?

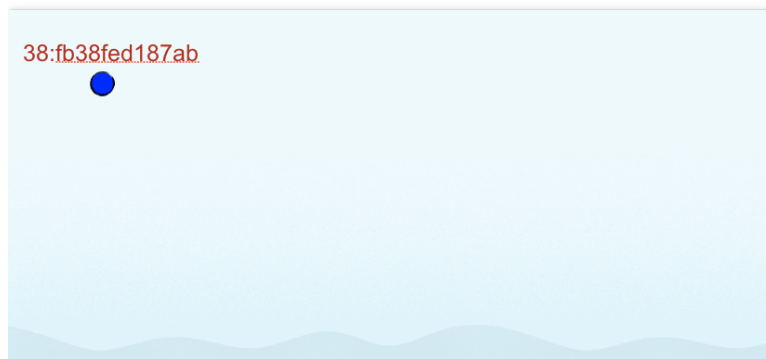


Réserve ...

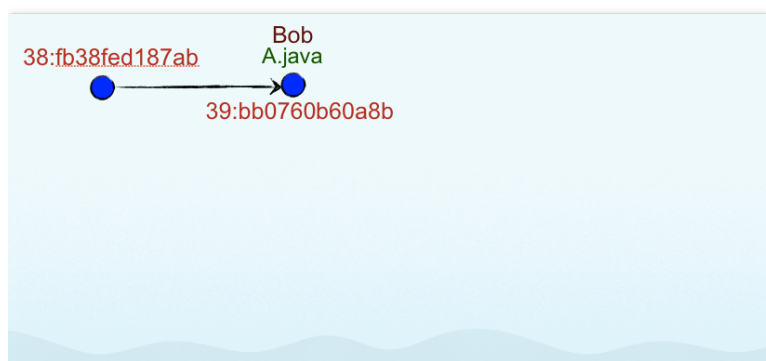
DVCS

- 'D' : pour *Distributed*

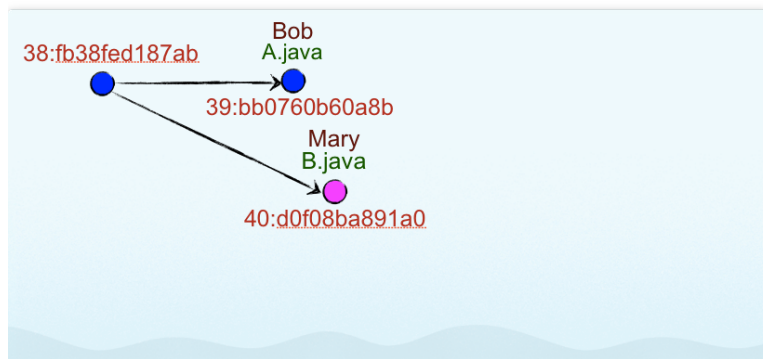
DAG



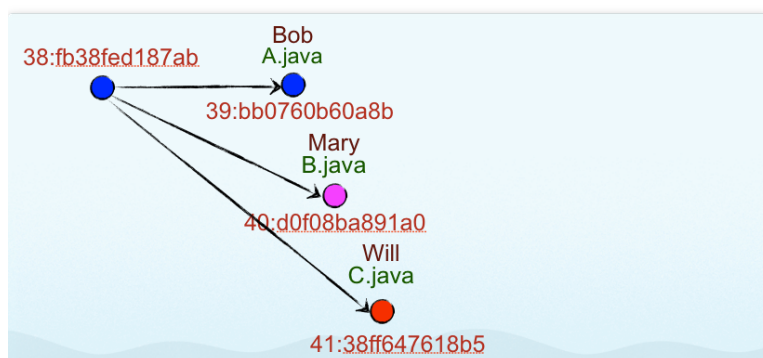
DAG



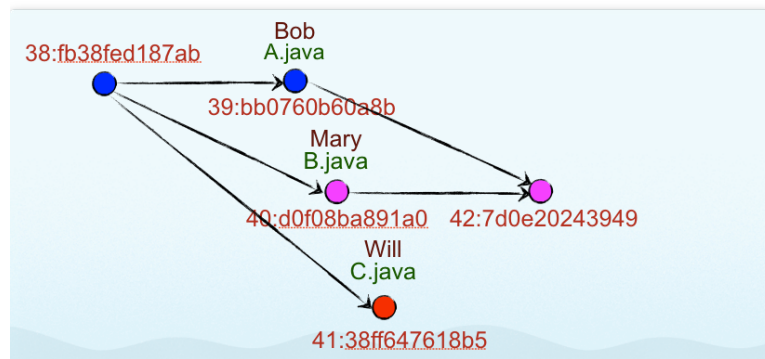
DAG



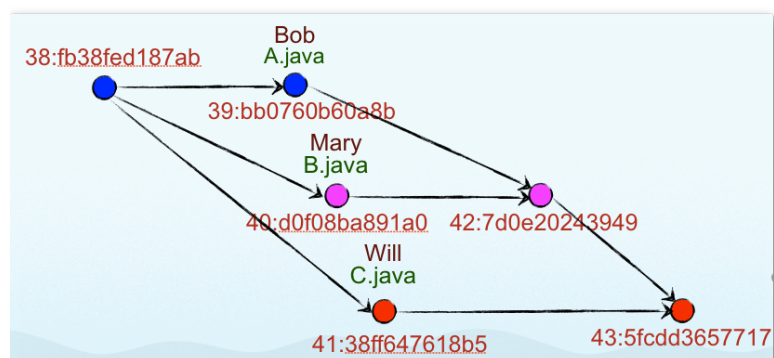
DAG



DAG



DAG

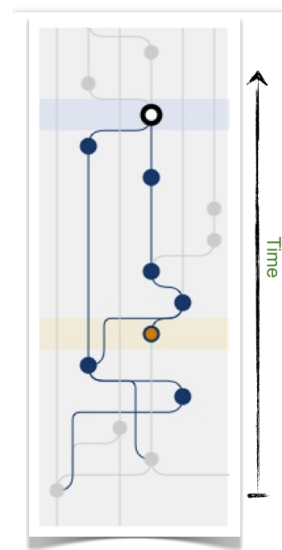


DVCS

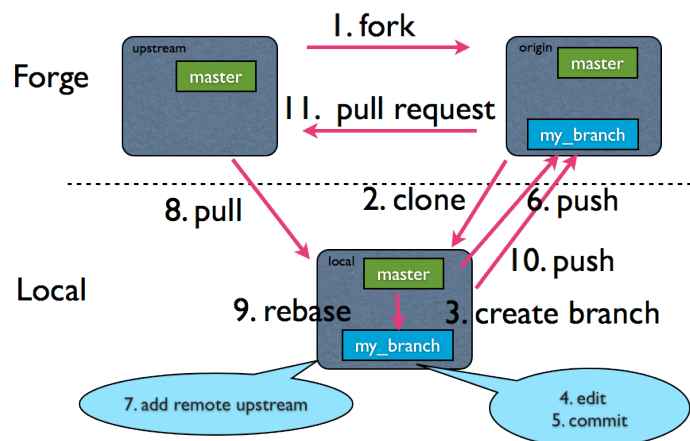
- Avantages :
 - déterministe
 - fusion/merge facile et efficace
 - gestion des branches simple et efficace

Traçabilité

- dans SVN ou CVS, les *merges* masquent ce qui s'est passé
- dans un DVCS, le DAG permet de savoir ce qui s'est passé exactement :
on a le détail de l'historique



Détails Pull-Request



Crédits

Les différentes images utilisées dans cette présentation sont tirées des pages suivantes :

- [Vincent Driessen](#)
- [Adam Dymitruk](#)
- [Scott Shacon](#)
- [gitflow](#)
- [hgflow](#)
- [git_bpf](#)
- workflow Pull Request [Atlassian](#)
- [SourceTree](#)
- [technical-debt.org](#)
- [Migrating to DVCS: Why You should do it and how Atlassian's tools can help](#)