

Introduction à la Gestion de Version

Licence Professionnelle IDSE
2015-2016

IUT de Nice

Simon Urli
urli@i3s.unice.fr

Rôles de la gestion de version

- Travailler en équipe sur un code commun
- Monitorer les changements
- Pouvoir revenir en arrière

Systemes de version centralisés

- Un seul dépôt central
 - Tous les utilisateurs «commit» sur ce dépôt
 - Tous les utilisateurs récupèrent les modifications du dépôt
- ▶ CVS, SVN, ...

Systemes de version décentralisés

- Autant de dépôts que d'utilisateurs
- Mais des dépôts hébergés sur des serveurs (ex: GitHub, BitBucket, ...)
- Un commit ne signifie PAS le partage !
- Obligation de faire commit ET push !
 - ▶ Git, Mercurial, ...

Avantages du décentralisé

- Mode «déconnecté» : possibilité de travail en local
- Prise en charge des branches beaucoup plus évoluée
- Plus fiable car serveur central est seulement une AUTRE copie des versions
- Beaucoup de plateformes le supporte

ATTENTION

- GitHub n'est PAS Git !
- GitHub est une **plateforme d'hébergement** de dépôts Git offrant en plus la possibilité d'annoter le code etc.

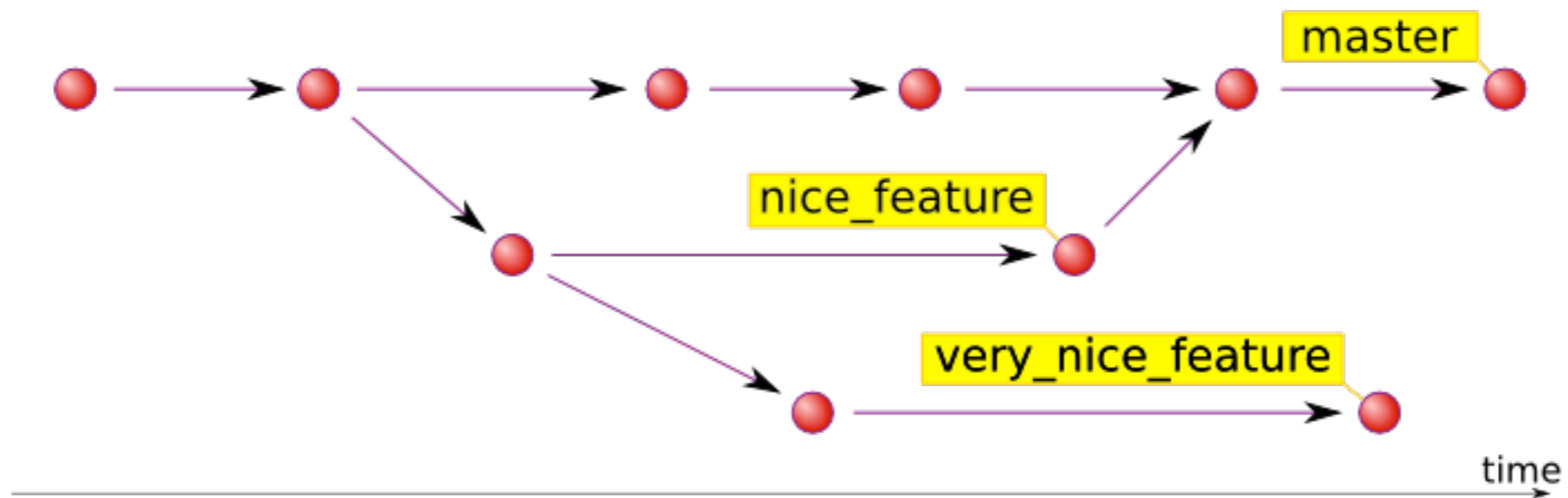
Quelques commandes de Git

- **init** : initialisation d'un dépôt vide
- **clone** : récupération d'une copie d'un dépôt
- **add** : ajout d'un fichier nouveau ou modifié pour le commit
- **commit** : enregistrement des modifications sur le dépôt
- **push** : envoie les modifications sur un serveur
- **pull** : récupère les modifications d'un serveur
- **status** : voir l'état du repository
- **branch** : gérer les branches
- **checkout** : switcher sur une autre version / branche
- **log** : afficher les infos des précédents commits
- ... : <https://git-scm.com/>

Les branches ?

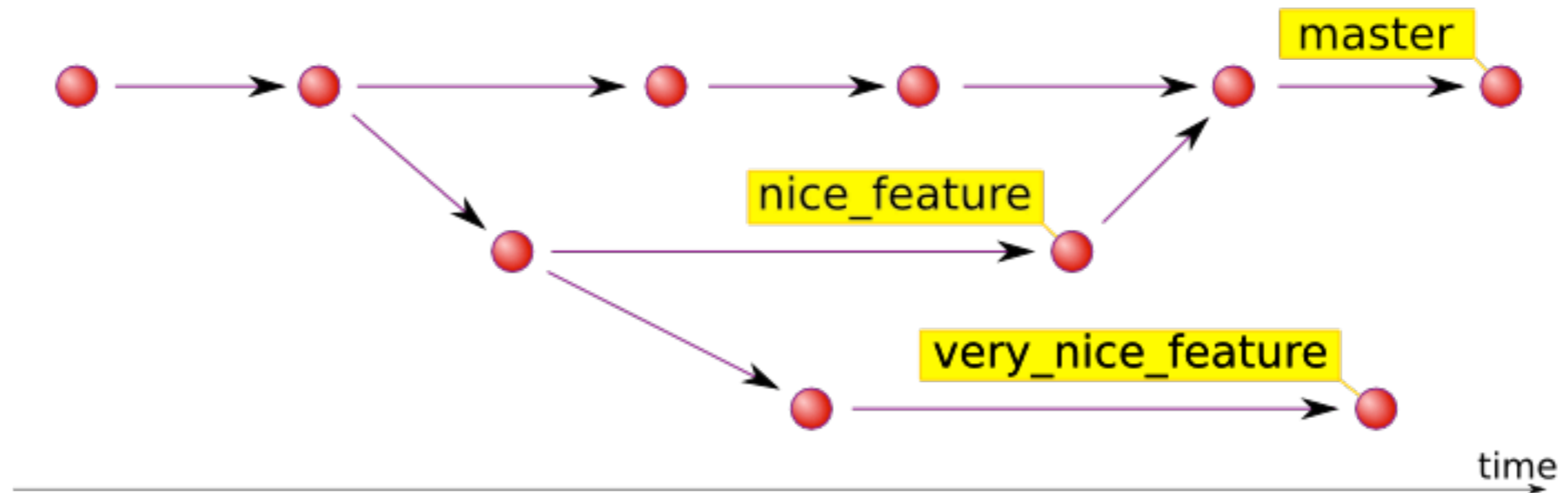


Les branches !



Chaque rond est un commit

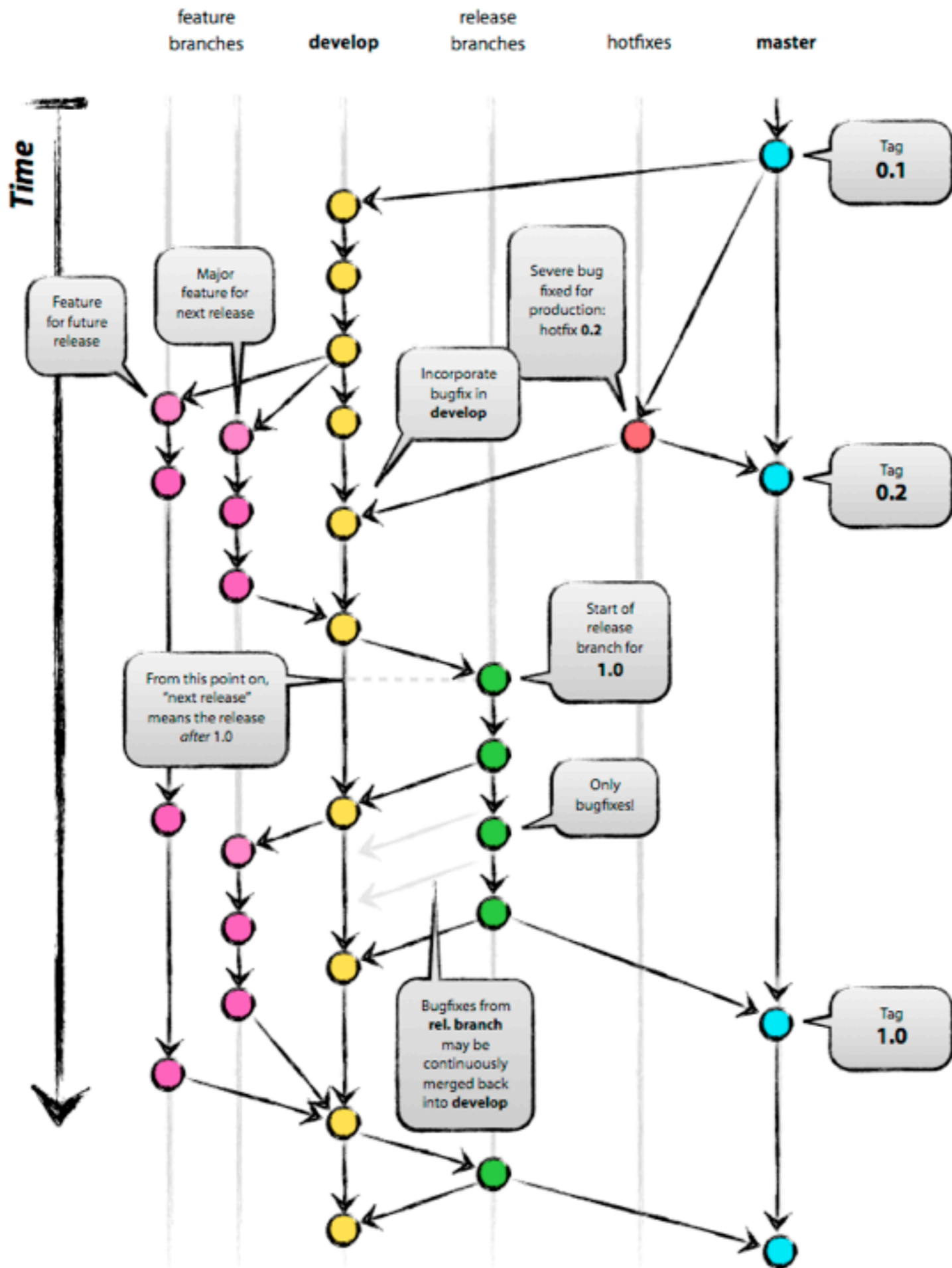
Les branches !



- Pouvoir travailler en parallèle sur plusieurs features en même temps
- Pouvoir switcher entre les features, les versions etc
- Fusionner les modifications sur une même branche à la fin

Git Flow : un processus d'utilisation de Git

- Idée de Git Flow : exploiter les branches au maximum !
- Principe : des branches partout !
 - Une branche « master » de releases
 - Une branche « develop » où tout se passe
 - Des branches « features » par fonctionnalité
 - Des branches pour les patches...



Un outil pour Git Flow

- ... qui s'appelle «Git Flow»
- Un set de command permettant de créer automatiquement les branches en fonction de ce qu'on fait
- Et qui permet de fermer/fusionner les branches !
- Sucre syntaxique au dessus de git !

Git Flow : quelques liens

- <http://nvie.com/posts/a-successful-git-branching-model/>
- <https://github.com/nvie/gitflow>