TACT factory
mobile agency

Fournisseur **souple** et **réactif** d'applications mobiles innovantes avec une **méthodologie** projet **rigoureuse**.

Présentation

Mickael Gaillard (Architecte logiciel)
mickael.gaillard@tactfactory.com

Yoan Pintas (Project manager)
yoan.pintas@tactfactory.com

Erwan LeHuitouze (Lead Developper)
erwan.lehuitouze@tactfactory.com

Planning session 2

Concepts Avancés :

- Gestion des tags
- Gestion des branches
- Le Merge de branches
- Résolution de conflits
- Centralisation

Rappel sur Git :

- git init
- git clone
- git status
- git add/rm/mv
- git commit
- git log
- git checkout
- .gitignore

Les tags

Au livrable on tag une version...

de6fe24b3 = 1.0.0

Liste des tags :

git tag

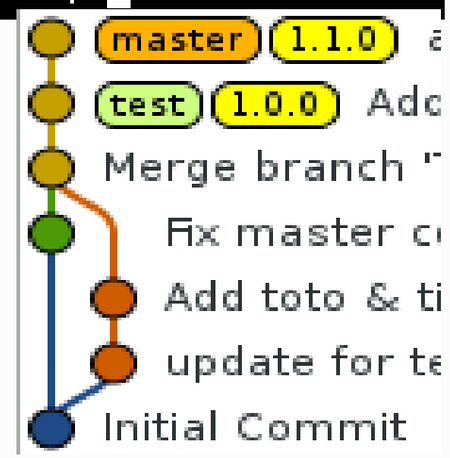
Créer un tag :

git tag <tag_name>

Basculer sur un tag :

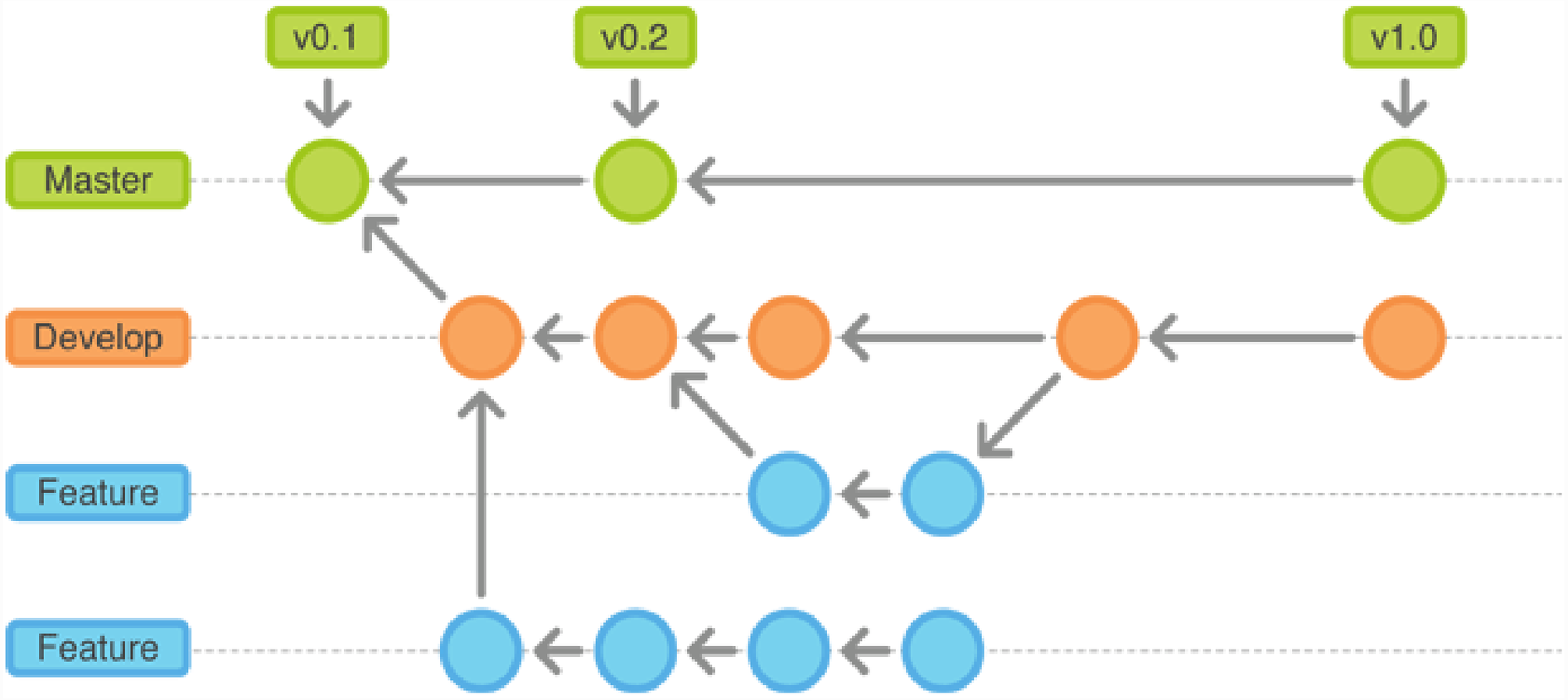
git checkout <tag_name>

```
micky@uranium:~/learning_git(master)$ git tag
1.0.0
micky@uranium:~/learning_git(master)$ git tag 1.1.0
micky@uranium:~/learning_git(master)$ git tag
1.0.0
1.1.0
micky@uranium:~/learning_git(master)$
```



Théorie des branches

Exemples :



Les branches

Liste des branches :

git branch

La branche courante est précédée de *

Créer une branche :

git checkout -b <branch_name>

Basculer dans une branche :

git checkout <branch_name>

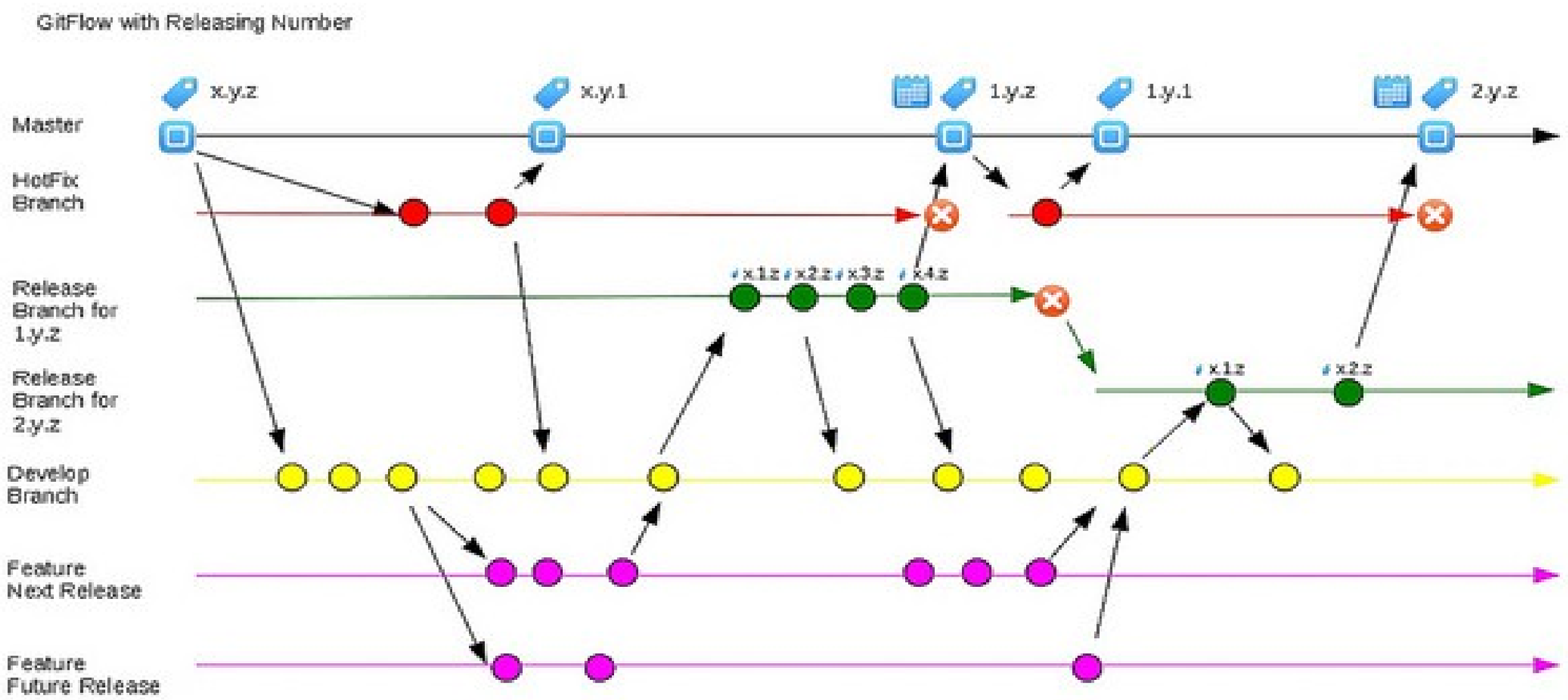
```
micky@uranium:~/learning_git(master)$ git branch
* master
micky@uranium:~/learning_git(master)$ git checkout -b test
Basculement sur la nouvelle branche 'test'
micky@uranium:~/learning_git(test)$ git branch
  master
* test
micky@uranium:~/learning_git(test)$ git checkout master
Basculement sur la branche 'master'
micky@uranium:~/learning_git(master)$ git branch
* master
  test
micky@uranium:~/learning_git(master)$
```

Supprimer une branche :

git branch -d <branch_name>

Le Merge de branches

Ré-assembler des branches

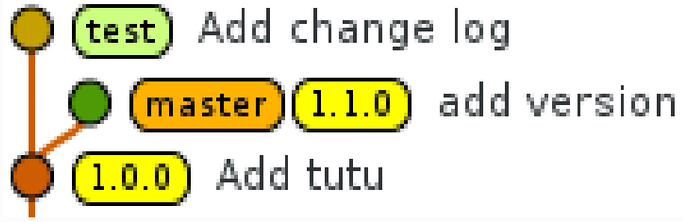


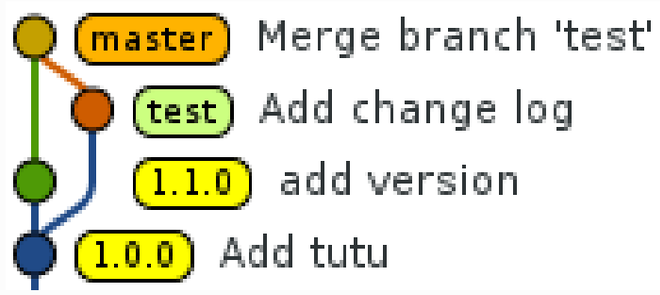
Le Merge de branches

Merge des branches :

git merge <branch_name>

Merge une branche dans la branche courante !!!





Rebaser une branche :

git rebase <branch_name>

Ne créer pas de commit !

```
micky@uranium:~/learning_git(test)$ git branch
  master
* test
micky@uranium:~/learning_git(test)$ git checkout master
Basculement sur la branche 'master'
micky@uranium:~/learning_git(master)$ git branch
* master
  test
micky@uranium:~/learning_git(master)$ git merge test
Merge made by the 'recursive' strategy.
 changelog.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 changelog.txt
micky@uranium:~/learning_git(master)$ git branch -d test
Branche test supprimée (précédemment ea023b0)
micky@uranium:~/learning_git(master)$
```

Résolution de conflits

Lors d'un merge...

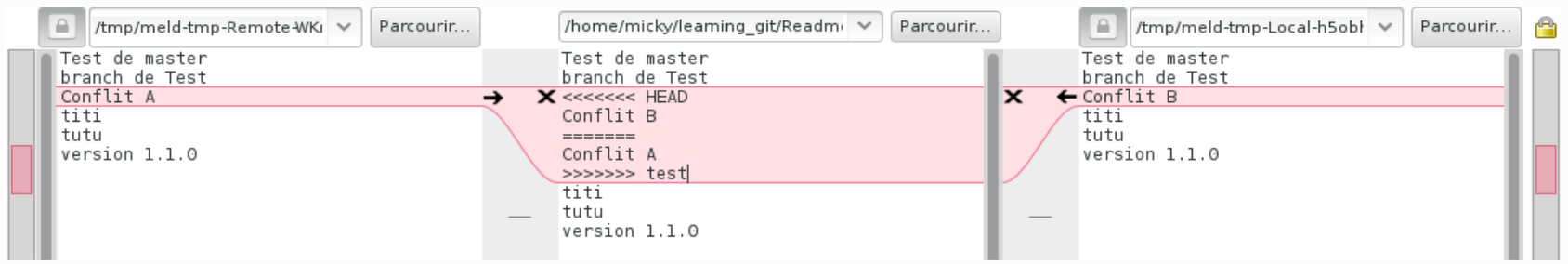
Structure d'un conflit :

<<<<<<< HEAD

=======

>>>>>>> test

```
micky@uranium:~/learning_git(master)$ git branch
* master
  test
micky@uranium:~/learning_git(master)$ git merge test
fusion automatique de Readme.txt
CONFLIT (contenu) : conflit de fusion dans Readme.txt
Automatic merge failed; fix conflicts and then commit the result.
micky@uranium:~/learning_git(master *+|MERGING)$ git add Readme.txt
micky@uranium:~/learning_git(master +|MERGING)$ git commit
[master ff510b2] Merge branch 'test'
micky@uranium:~/learning_git(master)$
```



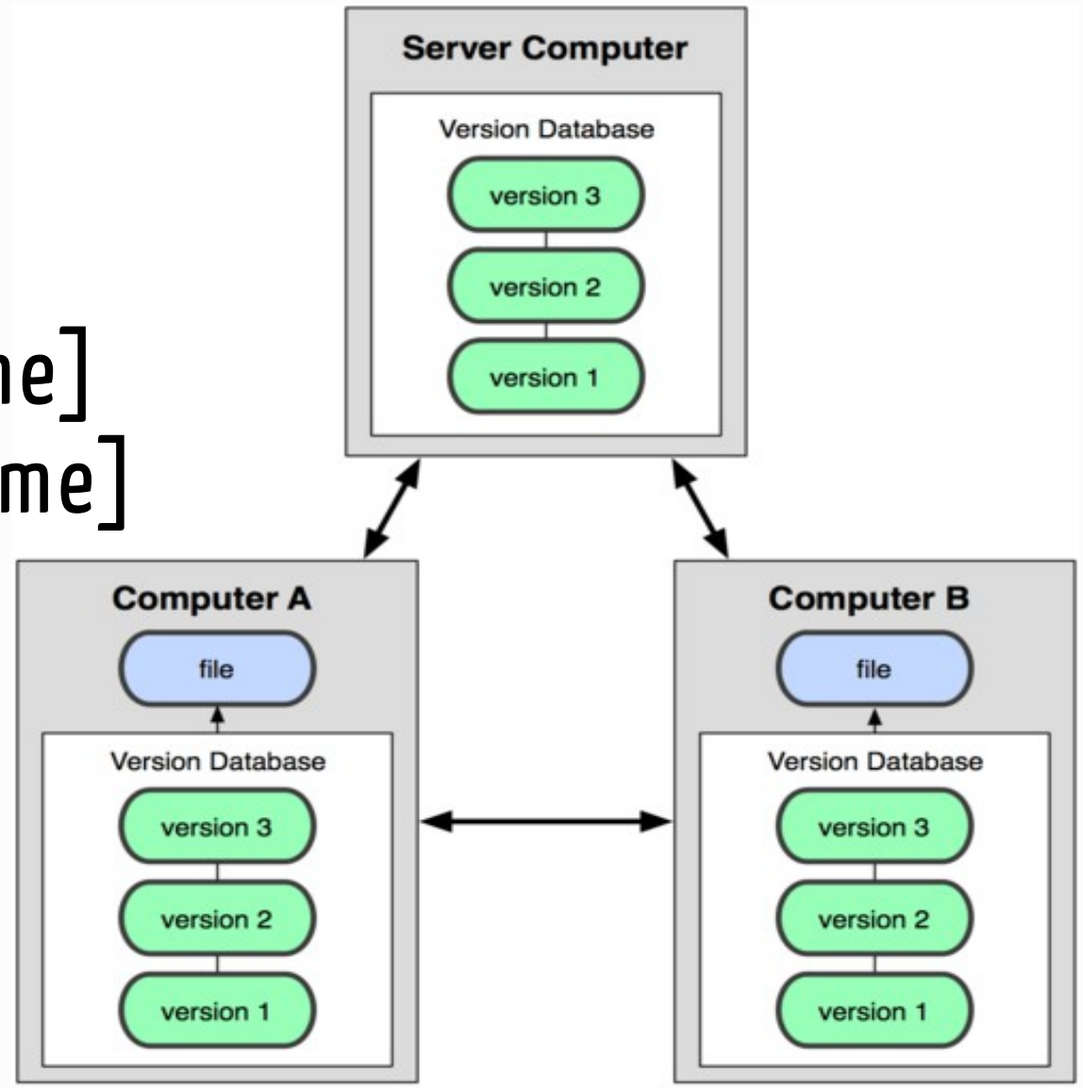
Pensez à ajouter et commiter la correction !

Centralisation

Utiliser SSH ! Mais http possible...

Synchroniser la database :

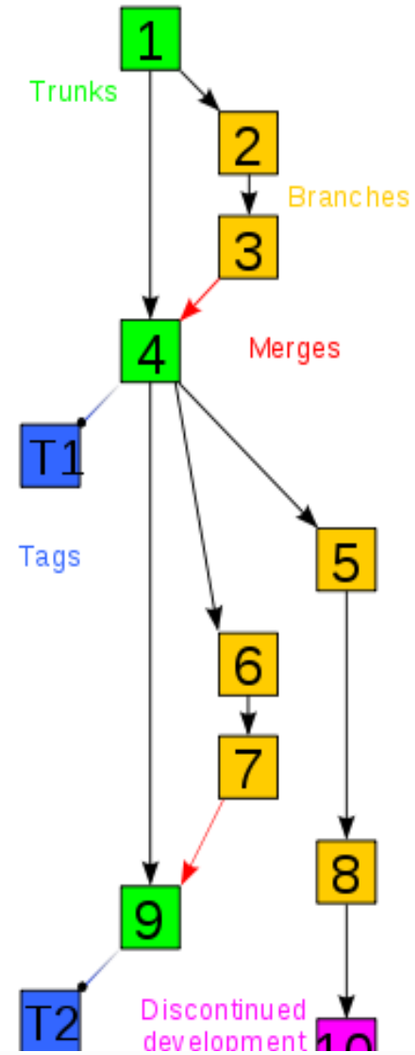
git pull [server_alias] [branch_name]
git push [server_alias] [branch_name]



Workflow de Git

Pas de règles mais des recommandations :

- Toujours créer une branche
- Faire de petit commit (et régulièrement)
- Avoir toujours du code fonctionnel (et/ou compilable)
- Puller avant de pusher
- Tagger les livrables
- Définir vos conventions de codage !!!



Questions ?

Change log

- Mickael Gaillard 2013 : Initial document