Git un système de gestion de versions

ALBERT SHIH

Direction Informatique
Observatoire de Paris

Journées Mathrice
16 mars 2016

Git

- Système de gestion de versions.
- Lancé par Linus Torvalds.
- Largement utilisé, GitHub.
- Beaucoup de documentation.
- *Truc de geek*.

En vrac

- Gestion de version dé-centraliser.
- git $\neq$ subversion
- La doc `man git-XXXX` par exemple `man git-add`
- git est bavard
- Intégration dans le shell (bash/zsh).
- Intégration dans emacs/vi/eclipse/etc. . . .
- Un dépot `git` $\longrightarrow$ `.git`.

Start your engine..

- `export $MES_REPOS='CEQUEVOUSVOULEZ'`
- `mkdir $MES_REPOS`
- `cd $MES_REPOS`
- `git clone https://github.com/git/git.git > /dev/null 2>/dev/null &`
- `mkdir repos_1`
- `cd repos_1`
- `git init`
- `ls -al`

Configuration

- Une série de configuration global ou local.
  - `git config --global --list`
  - `git config --global --add user.name "VOTRE NOM"`
  - `git config --global --add user.email VOTRE_EMAIL`
  - `git config --global --add color.ui auto`
  - `git config --global --add core.editor /usr/local/bin/vim`
  - `git config --global --list`
- Si besoin changer `git config --global --replace-all`
- Possibilité `git config --local`

Ajout fichiers

- `cd $MES_REPOS/repos_1`
- Faites un `cp /etc/[a-e]* .` ou `cp` de ce que vous voulez (≥ 2)
- `git status`
- `git add *`
- `git commit`
- `vi/emacs unfichier`
- `git status`
- `git add unfichier`
- `git commit`
  Mettre un VRAI commentaire
- Faire plusieurs fois les `vi`, `git add`, `git commit`

Staged

- `vi/emacs unfichier`
- `git status`
- `git add unfichier`
- `vi/emacs unfichier`
- `git status`
- `git commit`
- `git status`
- `git add unfichier`
- `git commit`

Ajout fichiers partiel

- `vi/emacs unfichier`
  Faire $\geq$ 2 groupes de modifications
- `git status`
- `git add -p unfichier`
- `git commit`
- `git status`
- `git add -p unfichier`
- `git commit`
- `git status`

Différence

- `vi/emacs unfichier`
- `git diff`
- `git add unfichier`
- `git diff --staged` (ou `git diff --cached`)

Parcours logs

- `cd $MES_REPOS/git`
- `git log`
- `git log --oneline`
- `git log -p`
- `git log -p XXXX...XXXX`
- `git log --author="Hamano"`
- `git log --grep="forget"`
- `git log -S"function"`
- `git diff XXXX XXXX`
- `git diff XXXX XXXX -- FICHIER`

C'est toi ?

- `cd $MES_REPOS/git`
- `git blame utf8.c`

Voir en arrière

- `cd $MES_REPOS/repos_1`
- `git log --oneline`
- `git checkout XXXXX`
- `git checkout HEAD`
- `git checkout XXXXX FICHIER`
- `git checkout HEAD -- FICHIER`

Revenir en arrière

- `cd $MES_REPOS/repos_1`
- `git log --oneline`
- Annuler **un** commit `git revert XXXXX`.
- Revenir à une version `git revert HEAD...XXXX`.
- Option `git revert -no-edit HEAD...XXXX` :
- `git log`

Revenir en arrière avec reset

- `cd $MES_REPOS/repos_1`
- `git log --oneline`
- `git reset XXXXX`
- `git log`
- Ne pas faire sur des données pusher.

Exemples

- Vos fichiers `TeX`, `Docs`, `.*`...
- Un arborecense `/var/www/spip`.
- Revenir en arrière avec les fichiers d'un puppet/salt/etc...

Conseils

- Un repos par contexte.
- Commiter régulièrements.
- Mettre de vrais commentaires (éviter `git commit -m`).
- Commiter régulièrements.
- Mettre de vrais commentaires (éviter `git commit -m`).

À plusieurs

- `cd $MES_REPOS/`
- `git clone repos_1 repos_2`
- `cd $MES_REPOS/repos_1`
- Faites quelques modifs dans `repos_1` et commiter
- `cd $MES_REPOS/repos_2`
- `git pull`
- `git log`
- Faites quelques modifs dans `repos_2` et commiter
- `git remote -v`

- `cd $MES_REPOS/repos_1`
- `git remote add repos2 $MES_REPOS/repos_2`
- `git pull`
- `git pull repos2 master`
- Faites quelques modifs dans `repos_2` et commiter
- `cd $MES_REPOS/repos_1`
- `git pull repos2 master`

- `cd $MES_REPOS`
- `git clone git@git.math.cnrs.fr:plm/LOGIN/repos_distant` (`git init --bare repos_distant`)
- `cd repos_1`
- `git remote add distant git@git.math.cnrs.fr:plm/LOGIN/repos_distant` (`git remote add distant file:///$MES_REPOS/repos_distant`)
- `git remote -v`
- `git push distant`
- `cd $MES_REPOS/repos_distant`
- `git pull`

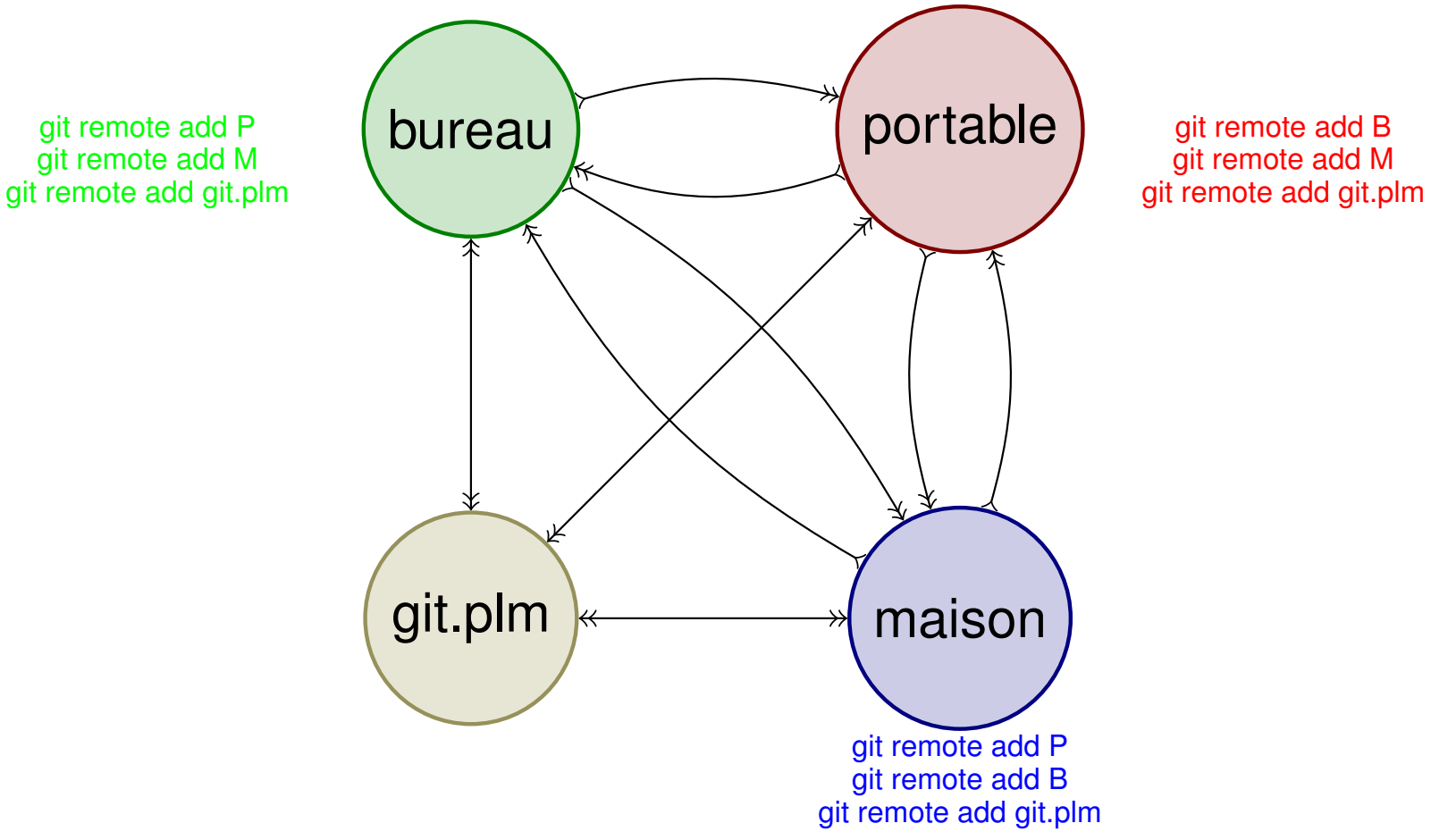
- `cd $MES_REPOS/repos_2`
- `git remote add repos1 $MES_REPOS/repos_1`
- `git remote add distant git@git.math.cnrs.fr:plm/LOGIN/repos_distant`
- `git remote -v`
- Faites quelques modifs dans `repos_2` et commiter
- `git push distant`
- `git push repos1`
- `cd $MES_REPOS/repos_1`
- `git pull repos2 master`

Conflits

- `cd $MES_REPOS/repos_1`
- `git pull repos2 master`
- `cd $MES_REPOS/repos_2`
- `git pull repos1 master`
- Faites quelques modifs dans `repos_1` et commiter
- Faites similaires modifs dans `repos_2` et commiter
- `cd $MES_REPOS/repos_2`
- `git pull repos1 master`
- `git status`
- Résoudre les conflits
- `git add FICHIER`
- `git commit`
- `git status`

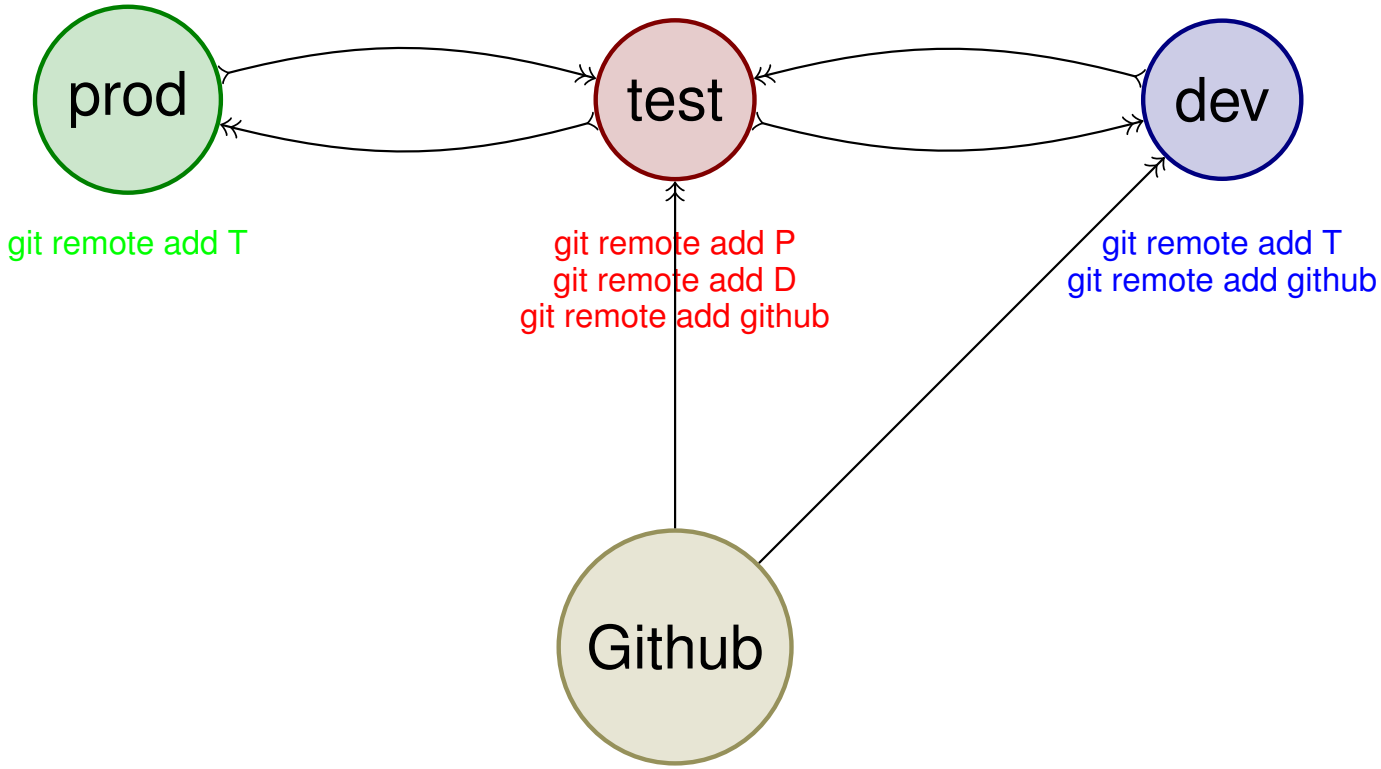
Home-dir

Votre home-dir



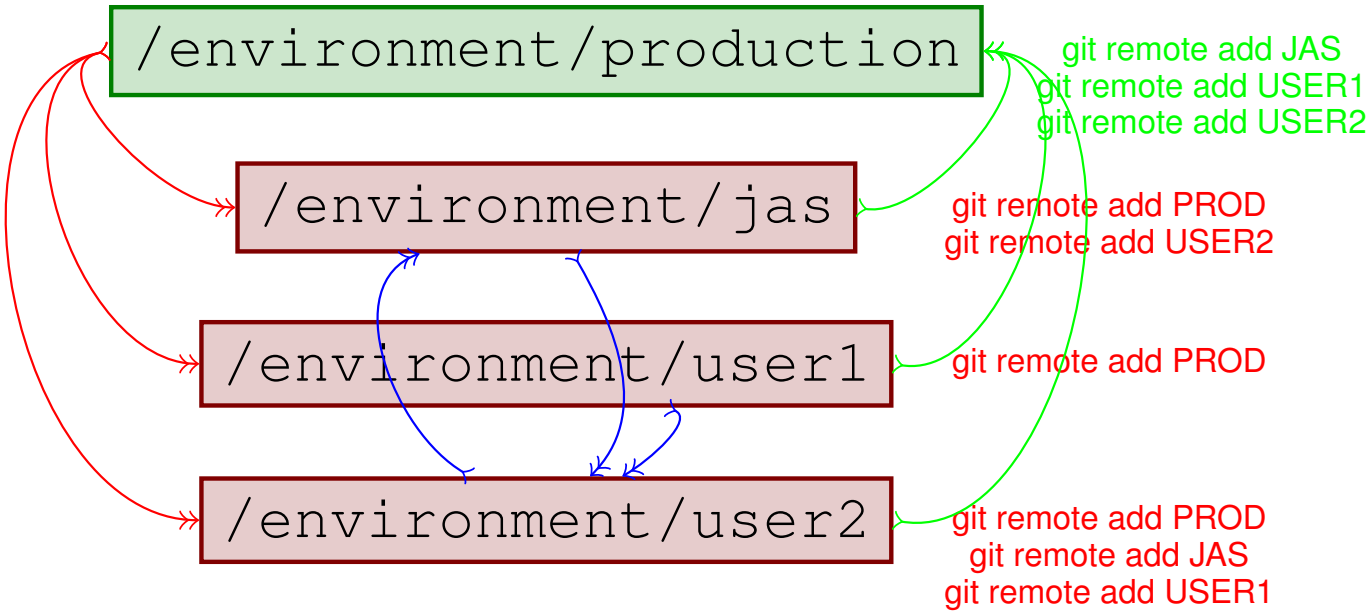
Appli PHP web

Votre `/var/www/spip`



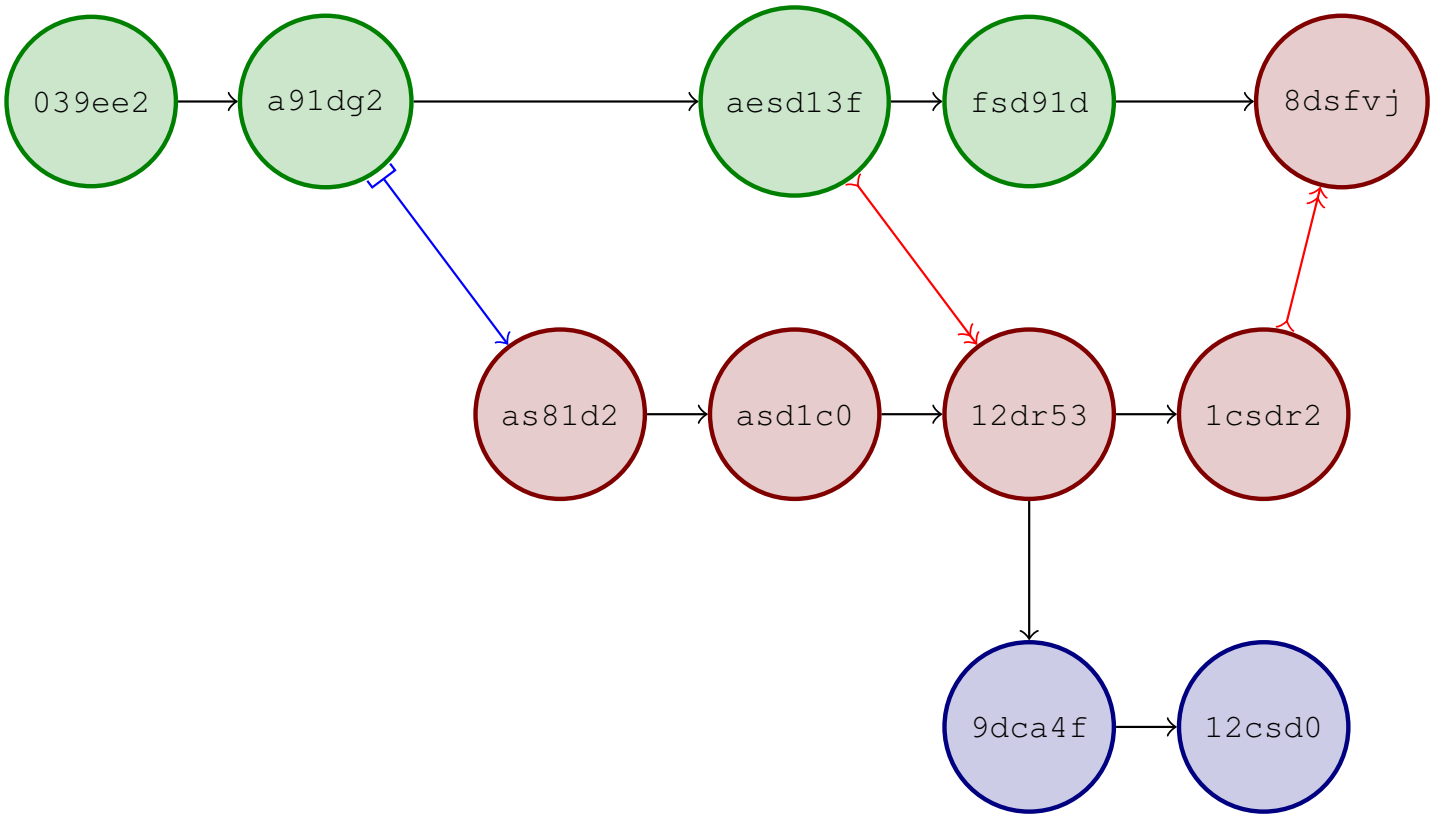
Puppet

Votre `/etc/puppet/environment/production`



- `cd /etc/puppet/environment/jas`
- `git pull`
- `vi *pp`, `git commit`, test, etc...
- `cd /et/puppet/environment/production`
- `git pull jas master`

Branches


039ee2
a91dg2
aesd13f
fsd91d
8dsfvj
as81d2
asd1c0
12dr53
1csdr2
9dca4f
12csd0

Branches

- `cd $MES_REPOS/repos_1`
- `git branch -l`
- `git checkout -b essai`
- `git branch -l`
- Faites des modifs dans et commiter plusieurs fois
- `git checkout master`
- `git merge essai`
- `git log --oneline -graph -decorate`
- Utiliser soit `tig` soit `gitk` pour visualiser les branches

Branches

- `cd $MES_REPOS/repos_1`
- `git branch -l`
- `git checkout essai`
- Faites des modifs et commiter plusieurs fois
- `git checkout master`
- Faites des modifs dans un fichier différent et commiter plusieurs fois
- `git merge essai`
- `git log --oneline -graph -decorate`
- Utiliser soit `tig` soit `gitk` pour visualiser les branches

Les branches et les remotes

- `cd $MES_REPOS/repos_2`
- `git fetch`
- `git branch -l`
- `git checkout -b monessai origin/essai`
- `cd $MES_REPOS/repos_1`
- `git checkout essai`
- Faites des modifs et commiter plusieurs fois
- `cd $MES_REPOS/repos_2`
- `git pull`
- Faites des modifs et commiter plusieurs fois

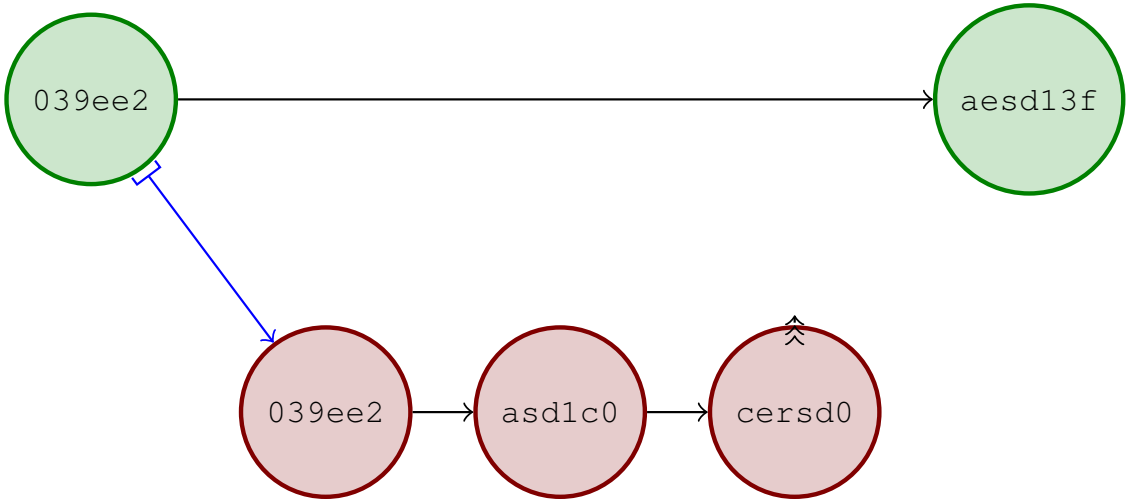
Les branches et les remotes

- `cd $MES_REPOS/repos_1`
- `git branch -a`
- `git fetch repos2`
- `git branch -a`
- `git checkout essai`
- `git branch --set-upstream-to=repos2/monessai essai`
- `git pull`
- `git log --oneline --decorate --graph` ou `tig` ou `gitk`

Appli PHP web

Mise à jour `/var/www/spip`

- `cd /var/www/spip`
- `git checkout -b 3.1.2`
- `cd /var/www/spip;unzip spip-3.1.zip`
- `git commit`
- Test.`git commit`
- `git checkout master; git merge -no-ff 3.1.2`
- `git branch -D 3.1.2`

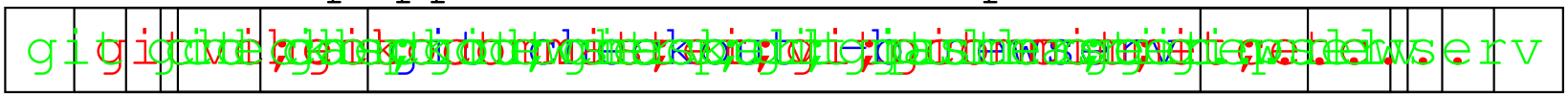


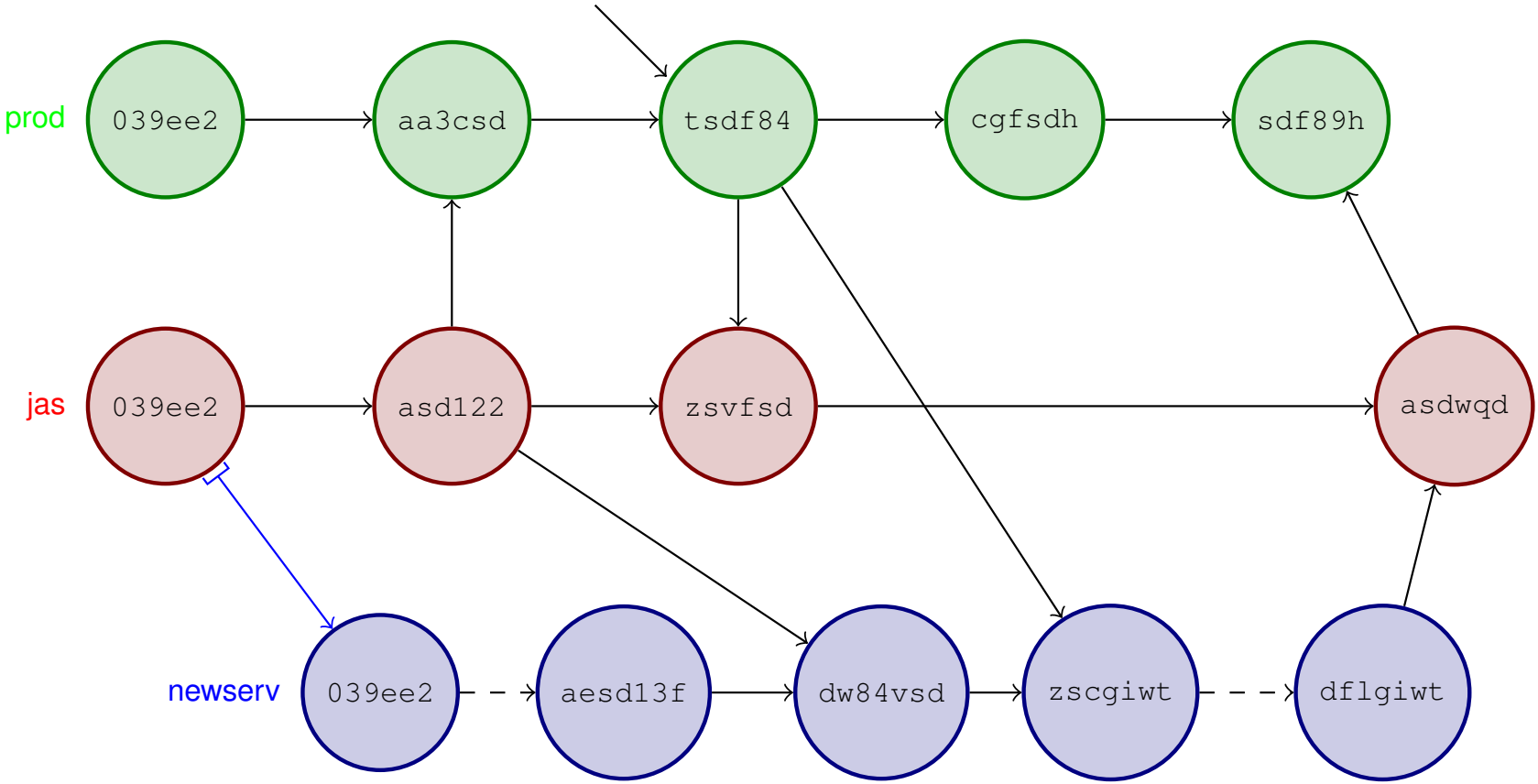
Branche vs revert

- Beaucoup plus souple pour basculer.
- Plus facile pour les diff.
- Historique plus propre.
- Plus facile à mettre en place sur différentes machines.

Puppet

Votre `/etc/puppet/environment/production`





stash

- `cd $MES_REPOS/repos_1`
- Faites des modifs (sans commit)
- `git stash`
- Faites des modifs et commiter.
- `git stash pop`
- Faites des modifs et commiter.

Historique

Ne pas faire si vous avez déjà fait git push Ne pas faire dans master

- `cd $MES_REPOS/repos_1`
- `git checkout essai`
- Faites des modifs et commiter $\geq$ 3 fois
- `git log`
- `git checkout nettoyage`
- `git rebase -i HEAD~3`
- `git log`
- `git checkout essai`
- `git merge nettoyage`
- `git checkout master`
- `git merge essai`
- `git log`

Tout droit ou pas (Fast foward)

- `cd $MES_REPOS/repos_1`
- `git pull repos2 master`
- `cd $MES_REPOS/repos_2`
- `git pull repos1 master`
- Faites quelques modifs dans `repos_1` et commiter
- `cd $MES_REPOS/repos_2`
- `git pull repos1 master`
- Utiliser soit `tig` soit `gitk` soit `git log --oneline --graph --decorate` pour visualiser les branches

Tout droit ou pas

- `cd $MES_REPOS/repos_1`
- `git pull repos2 master`
- `cd $MES_REPOS/repos_2`
- `git pull repos1 master`
- Faites quelques modifs dans `repos_1` et commiter
- Faites quelques modifs (sans conflit) dans `repos_2` et commiter
- `cd $MES_REPOS/repos_2`
- `git pull repos1 master`
- Utiliser soit `tig` soit `gitk` soit `git log --oneline --graph --decorate` pour visualiser les branches

Tout droit ou pas

- `cd $MES_REPOS/repos_1`
- `git pull repos2 master`
- `cd $MES_REPOS/repos_2`
- `git pull repos1 master`
- Faites quelques modifs dans `repos_1` et commiter
- Faites quelques modifs (sans conflit) dans `repos_2` et commiter
- `cd $MES_REPOS/repos_2`
- `git pull --rebase repos1 master`
- Utiliser soit `tig` soit `gitk` soit `git log --oneline --graph --decorate` pour visualiser les branches

À faire

- Un commit = petit diff.
- Commiter, commiter, . . .
- Utiliser les branches.
- Faites des **vrai** message de log.
- Ré-écrire l'historique.
- Faites des historiques propres.

À pas faire

- Beaucoup de modifications dans un commit.
- `git commit -m`
- Ne **jamais** faire `rebase -i` sur des données déjà pusher/puller.

Pour aller plus loin

- `tags`
- `cherry-pick`
- `rerere`
- `bisect`
- `subtree`
- `submodules`

Merci pour votre attention