Git vs. SVN

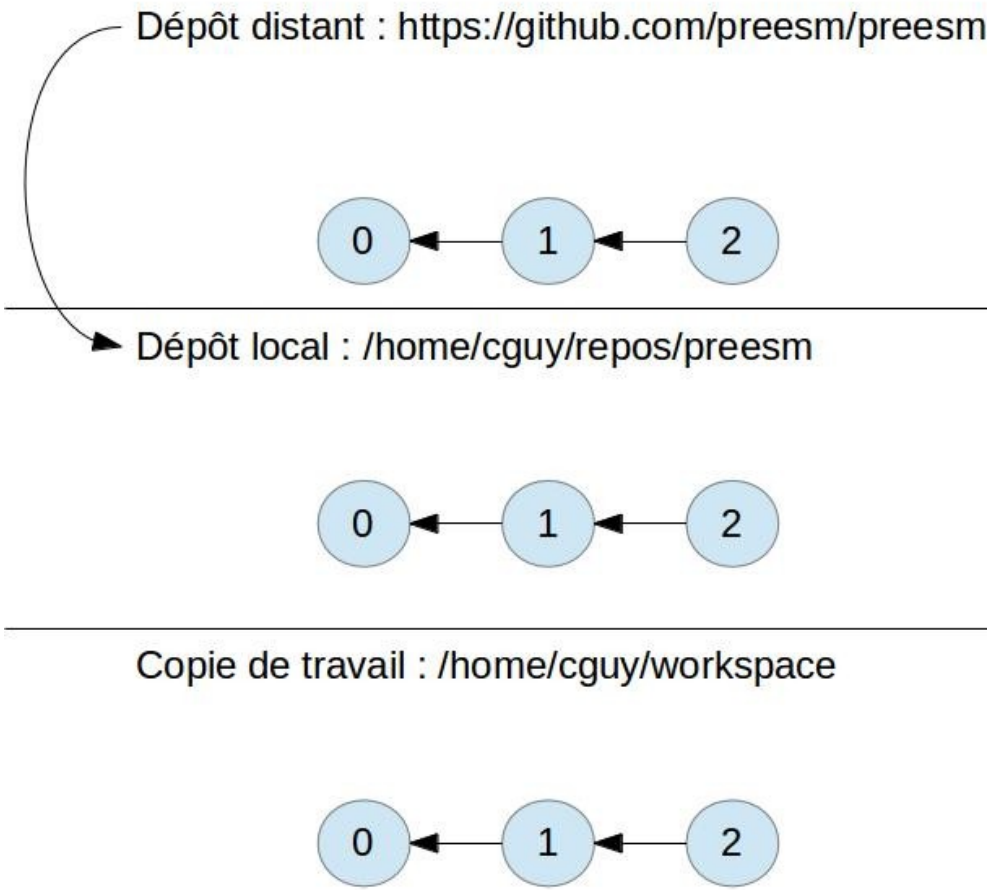
Décentralisé

- Pas de dépendance au réseau
- 1+ dépôt local/utilisateur (clone d'un dépôt distant)
- 1 étape supplémentaire
  - commit (local) + push (distant)

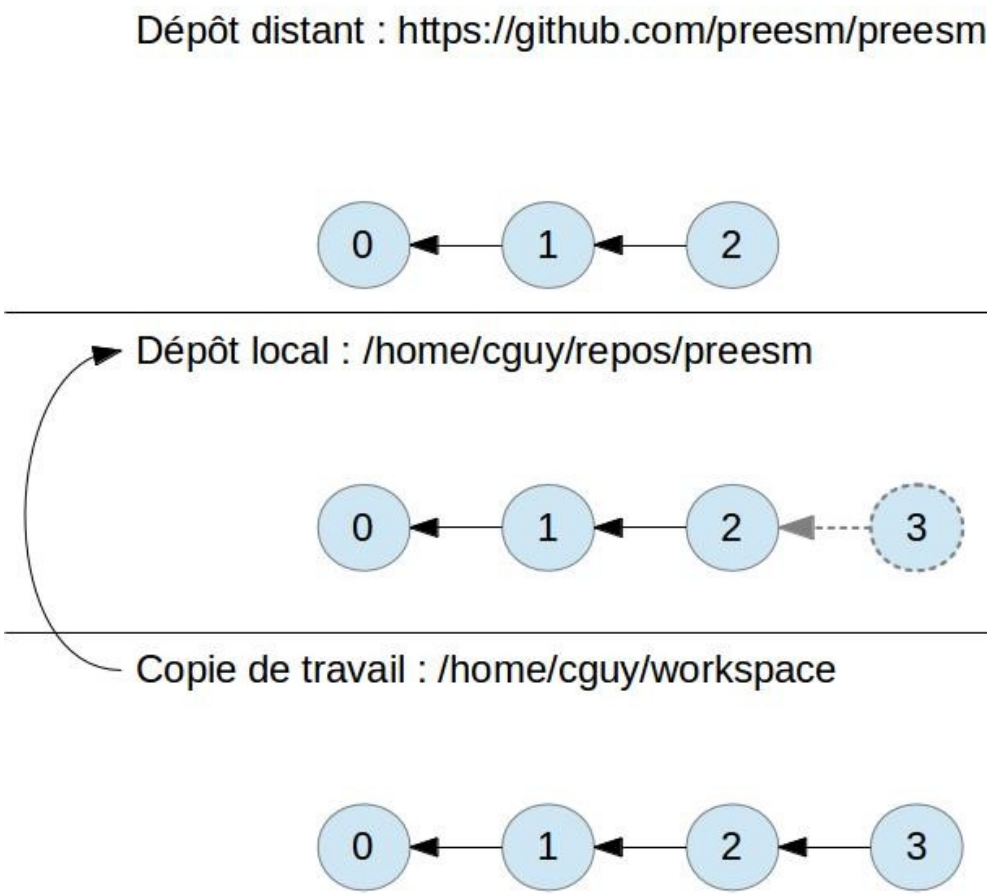
Gestion des branches

- Tags
- Légère, simple, rapide

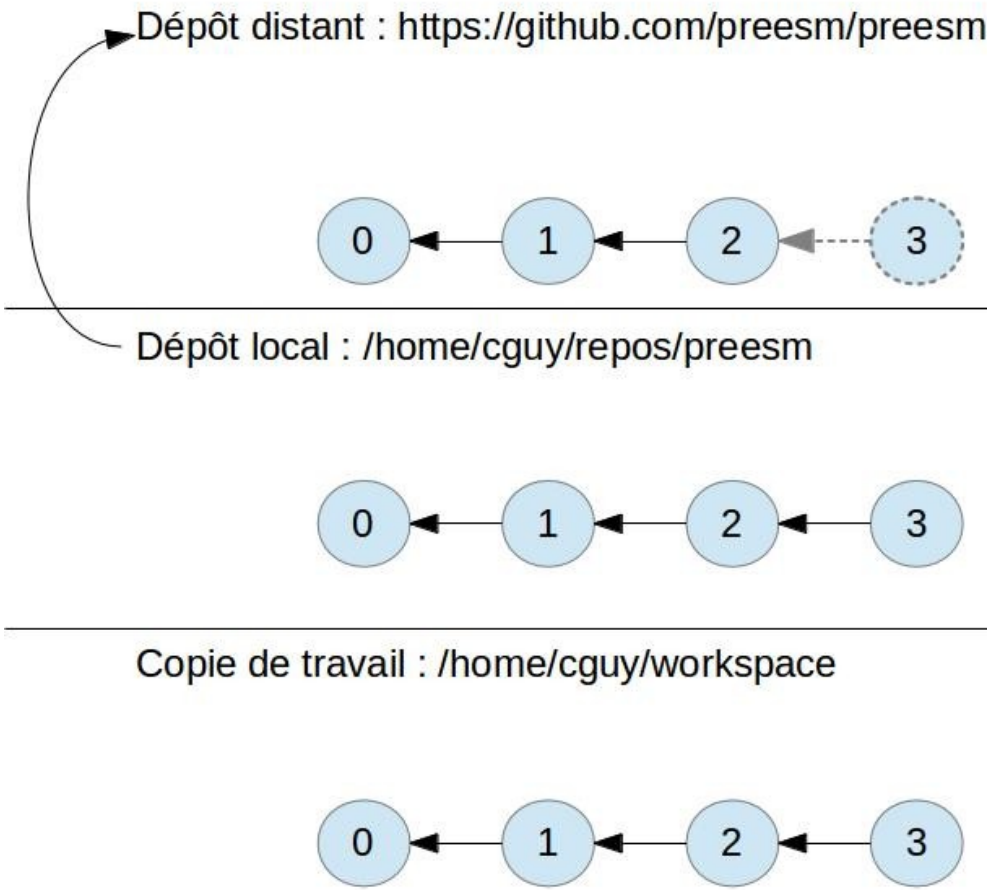
Clone


Dépôt distant : https://github.com/preesm/preesm
0
1
2
Dépôt local : /home/cguy/repos/preesm
0
1
2
Copie de travail : /home/cguy/workspace
0
1
2

Commit


Dépôt distant : https://github.com/preesm/preesm
0
1
2
Dépôt local : /home/cguy/repos/preesm
0
1
2
3
Copie de travail : /home/cguy/workspace
0
1
2
3

Push


Dépôt distant : https://github.com/preesm/preesm
0
1
2
3
Dépôt local : /home/cguy/repos/preesm
0
1
2
3
Copie de travail : /home/cguy/workspace
0
1
2
3

Termes usuels

`master`

Branche principale

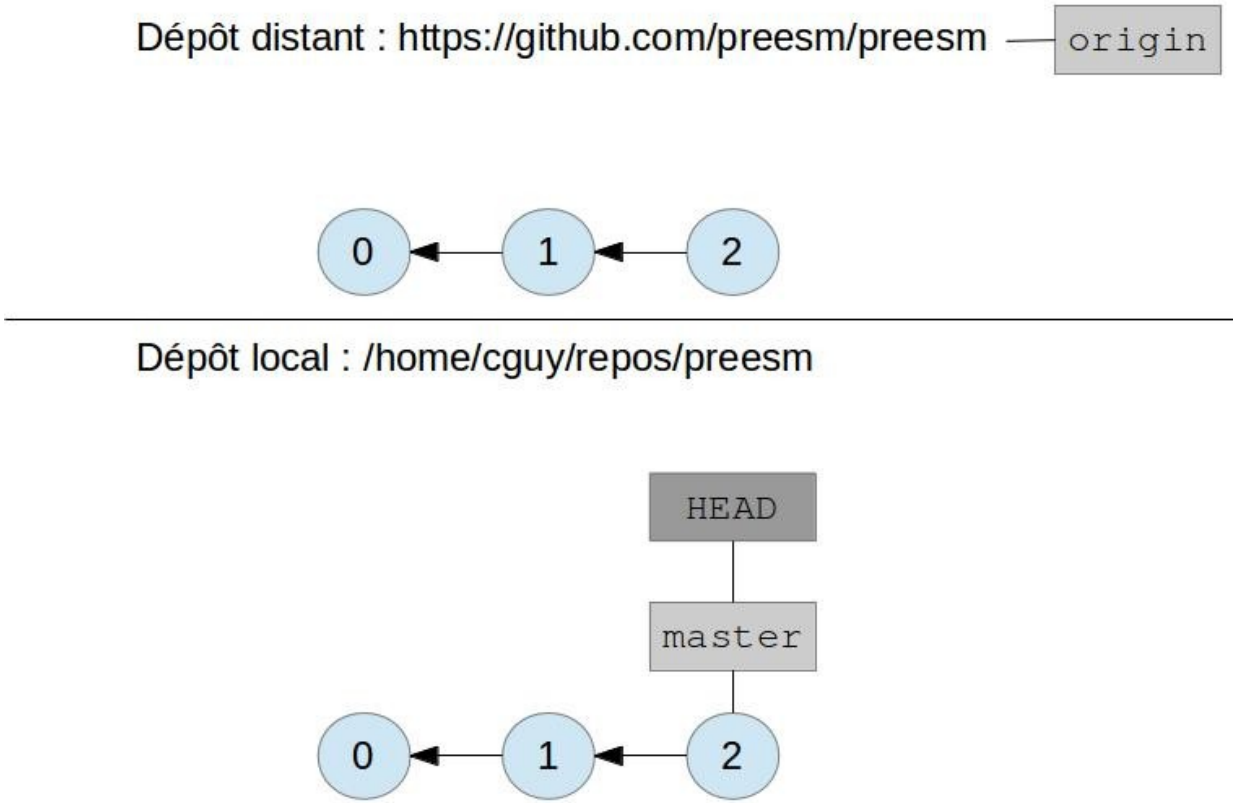
`HEAD`

Pointeur sur le dernier commit de la branche courante

`origin`

Dépôt distant qui a été cloné pour créer le dépôt local

Termes usuels



Commandes Git

Version Control Layer	Local commands	**add** annotate apply archive bisect blame **branch** check-attr **checkout** cherry-pick **clean commit diff** filter-branch grep **help init log merge mv** notes rebase rerere **reset** revert **rm** shortlog show-branch **stash status** submodule **tag** whatchanged
	Sync with other repositories	**am** bundle **clone** daemon fast-export fast-import **fetch format-patch** http-backend http-fetch http-push imap-send mailsplit **pull push** quiltimport **remote** request-pull send-email shell update-server-info
	Sync with other VCS	archimport cvsexportcommit cvsimport cvsserver **svn**
	GUI	citool **difftool gitk gui** instaweb **mergetool**
VC Low-Level Layer		checkout-index check-ref-format cherry commit-tree **describe** diff-files diff-index diff-tree fetch-pack fmt-merge-msg for-each-ref fsck **gc** get-tar-commit-id ls-files **ls-remote** ls-tree mailinfo merge-base merge-file merge-index merge-one-file mergetool--lib merge-tree mktag mktree **name-rev** pack-refs parse-remotes patch-id prune read-tree receive-pack reflog replace rev-list rev-parse send-pack **show show-ref** sh-setup stripspace symbolic-ref update-index update-ref upload-archive **verify-tag** write-tree
Utilities		**config** var web--browse
Database Layer		cat-file count-objects hash-object index-pack pack-objects pack-redundant prune-packed relink repack show-index unpack-file unpack-objectsupload-pack verify-pack
Database (blobs, trees, commits, tags)		

Création de dépôt

`git init <repo>`

Initialise un dépôt git local (.git)

`git clone <url>`

Créé un dépôt local copie du dépôt distant

Gestion de l'index

Index (Staging area)

Contient les modifications qui seront committées au prochain commit

`git add <file>`

Ajoute l'état courant d'un fichier à l'index

`git rm <file>`

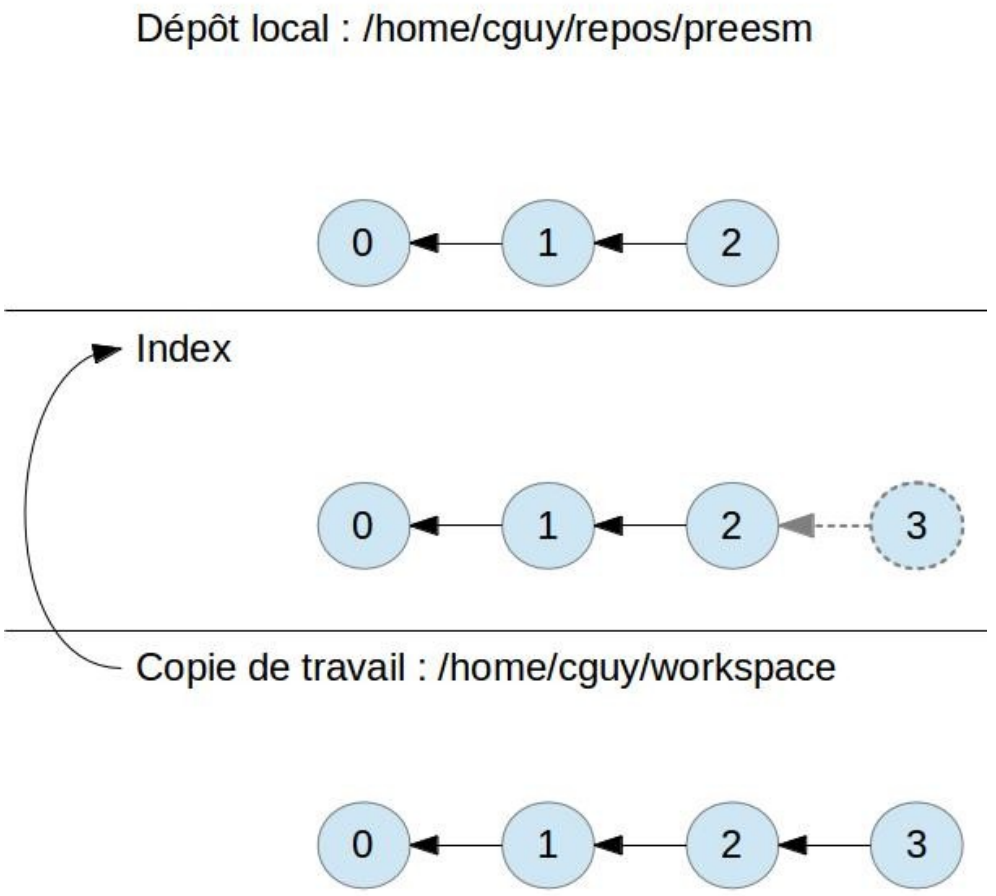
Supprime un fichier de l'index **et** de la copie de travail

`--staged` supprime un fichier de l'index

`git mv`

Déplace un fichier (équivalent `mv`, `git add` et `git rm`)

git add


Dépôt local : /home/cguy/repos/preesm
0
1
2
Index
0
1
2
3
Copie de travail : /home/cguy/workspace
0
1
2
3

Gestion des commits

`git commit [-m <message>]`

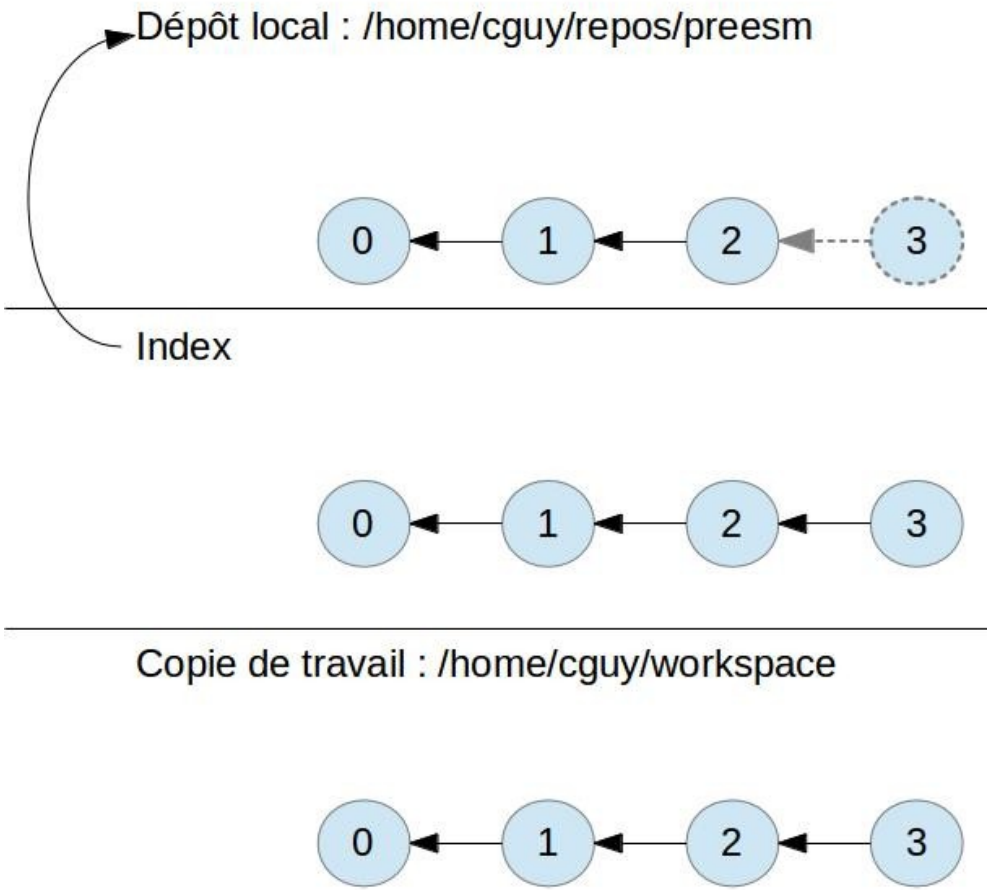
Committe les modifications indexées dans le dépôt **local**

`git reset`

Restaure l'index à l'état du dernier commit

`--hard` Restaure l'index **et** la copie de travail (équivalent à `svn revert`)

git commit


Dépôt local : /home/cguy/repos/preesm
0
1
2
3
Index
0
1
2
3
Copie de travail : /home/cguy/workspace
0
1
2
3

Afficher l'état du dépôt

`git status`

Affiche l'état de l'index et de la copie de travail

`git diff`

Affiche les différences entre l'index et la copie de travail

`--staged` Affiche les différences entre l'index et `HEAD`

`git gui`

`git status` avec GUI

`gitk`

Affiche l'historique de la branche courante

`--all` Affiche l'historique du dépôt

« TP » 1/3

Vérifier les changements à chaque fois avec `git status`, `git gui`, `gitk --all`

- Créer un dépôt local
- Créer un fichier `.gitignore`
  - L'indexer
  - Le committer
  - Ajouter `*~` dans le fichier
  - L'indexer
  - Le committer
- Créer un fichier et le committer
  - Le renommer et le committer (de deux manières)
  - Le supprimer et le committer
- Créer un fichier et le committer
  - Le modifier
  - Afficher les différences entre l'index et la copie de travail
  - L'indexer
  - Afficher les différences entre l'index et la copie de travail
- Restaurer l'index
- Restaurer la copie de travail

Gestion des branches

`git branch`

Affiche la liste des branches

`<branch>` crée une nouvelle branche à partir de la branche courante

`git checkout [-b] <branch>`

Bascule d'une branche à une autre

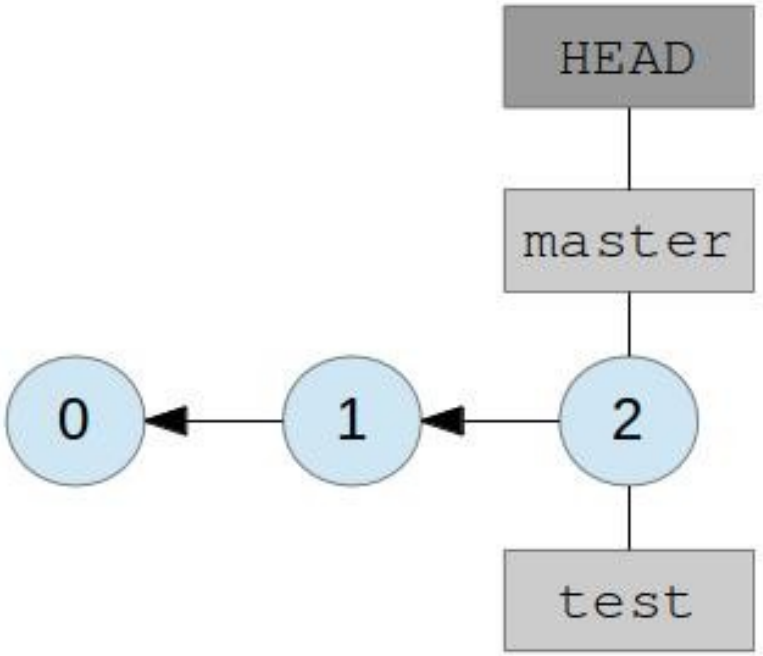
`-b` crée la branche branch avant de basculer dessus

`git stash`

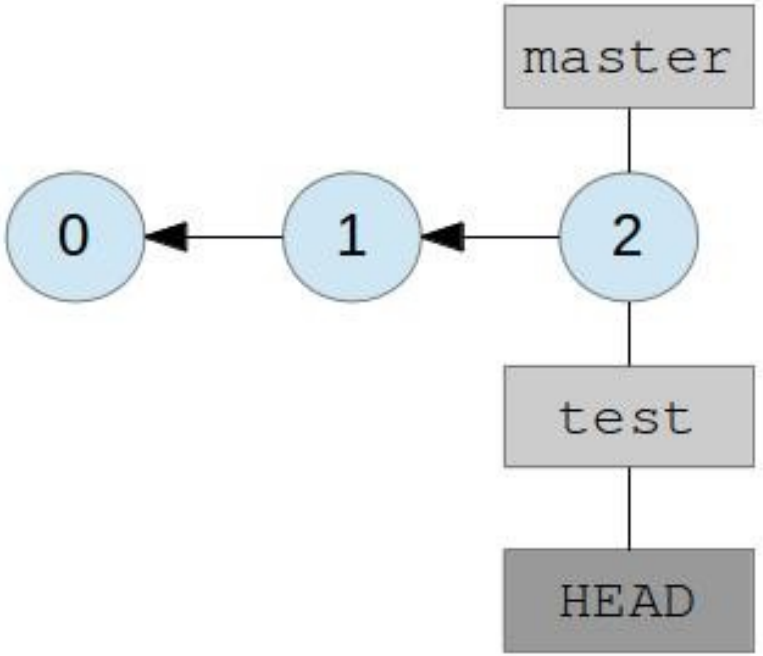
Remise les modifications non-committés

`apply` ré-applique les modifications remisées dans la branche **courante**

git branch test



git checkout test



Fusion de branches

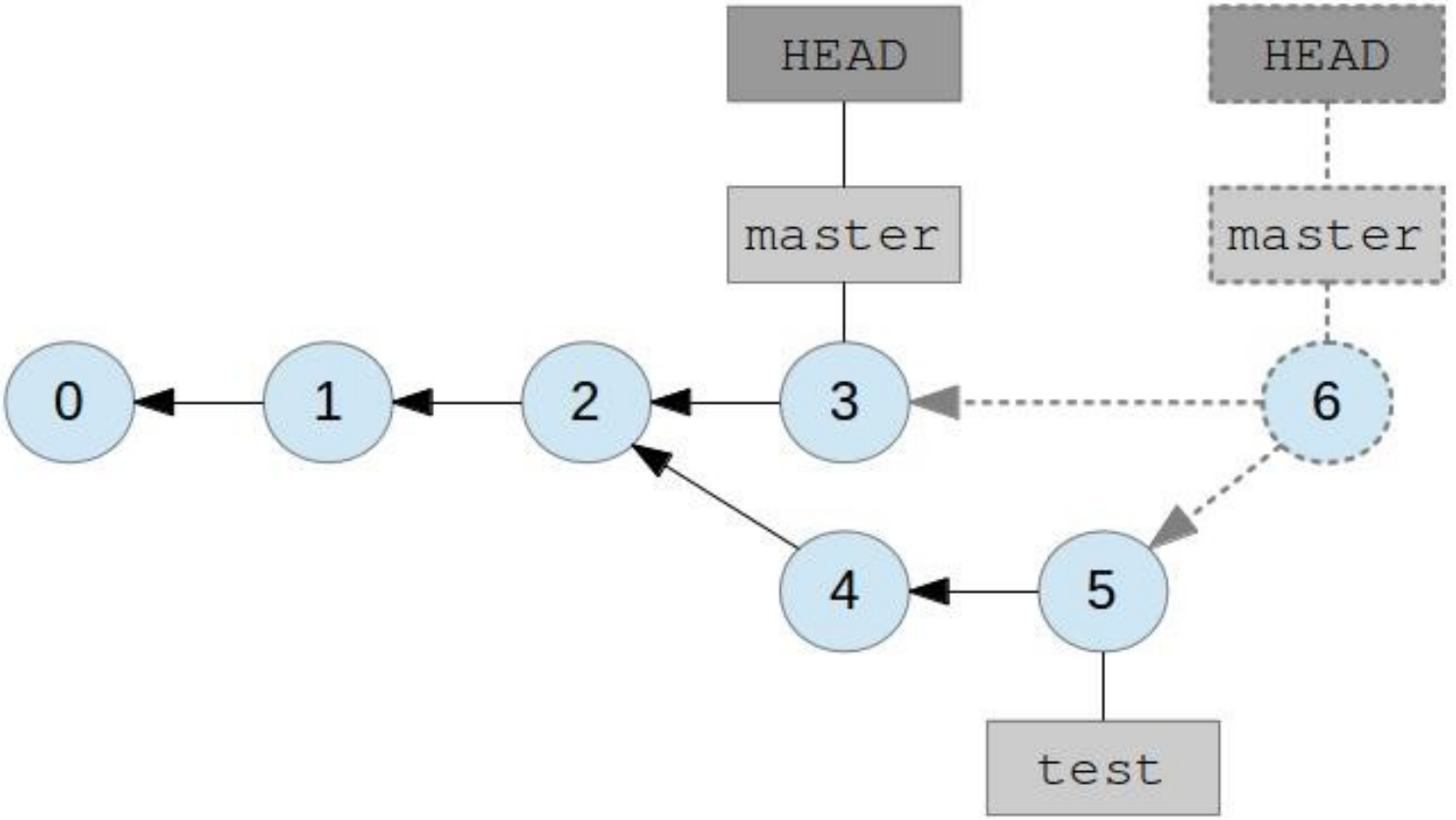
`git merge <branch>`

- Fusionne `branch` dans la branche courante en créant un nouveau commit

En cas de conflit, le commit est interrompu jusqu'à la résolution du conflit

- Édition des fichiers conflictuels
- Ajout à l'index
- Commit

git merge test



« TP » 2/3

Vérifier les changements à chaque fois avec `git status`, `git gui`, `gitk --all`

- Créer une branche `test` et basculer dessus
- Faire des modifications, les committer
- Basculer sur `master`
- Faire des modifications, les committer
- Fusionner `test` dans `master`
- Committer des modifications dans chaque branche pour créer un conflit (modifier deux fois la même partie d'un fichier)
- Fusionner `test` dans `master` et résoudre le conflit

Dépôts distants

`git remote`

Affiche la liste des dépôts distants enregistrés

`add <repo> <url>` ajoute un dépôt distant

`git fetch <repo>`

Récupère les données d'un dépôt distant (**sans fusion**)

`git pull`

Équivalent de `git fetch` + `git merge`

`git push`

Pousse les changements dans le dépôt distant

Suivi de branche (1)

1 branche locale peut suivre 1 branche distante

- Les `pull`/`push` se font **sur/depuis** cette branche
- Pas de suivi → pas de synchronisation

En général `master` suit `origin/master`

Suivi de branche (2)

`git checkout -b <localbranch> <repo>/<distantbranch>`

La branche locale `localbranch` suit la branche distante `distantbranch`

`git push [-u] <repo> <branch>`

Pousse la branche locale `branch` sur le dépôt distant `repo`

`[-u] branch` suit la branche distante créée

« TP » 3/3

Vérifier les changements à chaque fois avec `git status`, `git gui`, `gitk --all`

- Créer un compte Github
- Cloner le dépôt `https://github.com/ClementGGuy/toyrepo.git`
- Créer une branche à votre nom et basculer dessus
- Ajouter un fichier, l'indexer, le committer
- Pousser la branche
- Récupérer la branche d'un autre participant
- Basculer sur `master`
- Modifier la première ligne du fichier `common-files/conflicts-generator.txt`
- Le committer et le pousser
- Tirer la branche distante sans la fusionner
- Fusionner et résoudre le conflit

git-flow

2 branches principales et permanentes :

- `master` → Base de code (release actuelle)
- `develop` → Intégration features & bug fixes

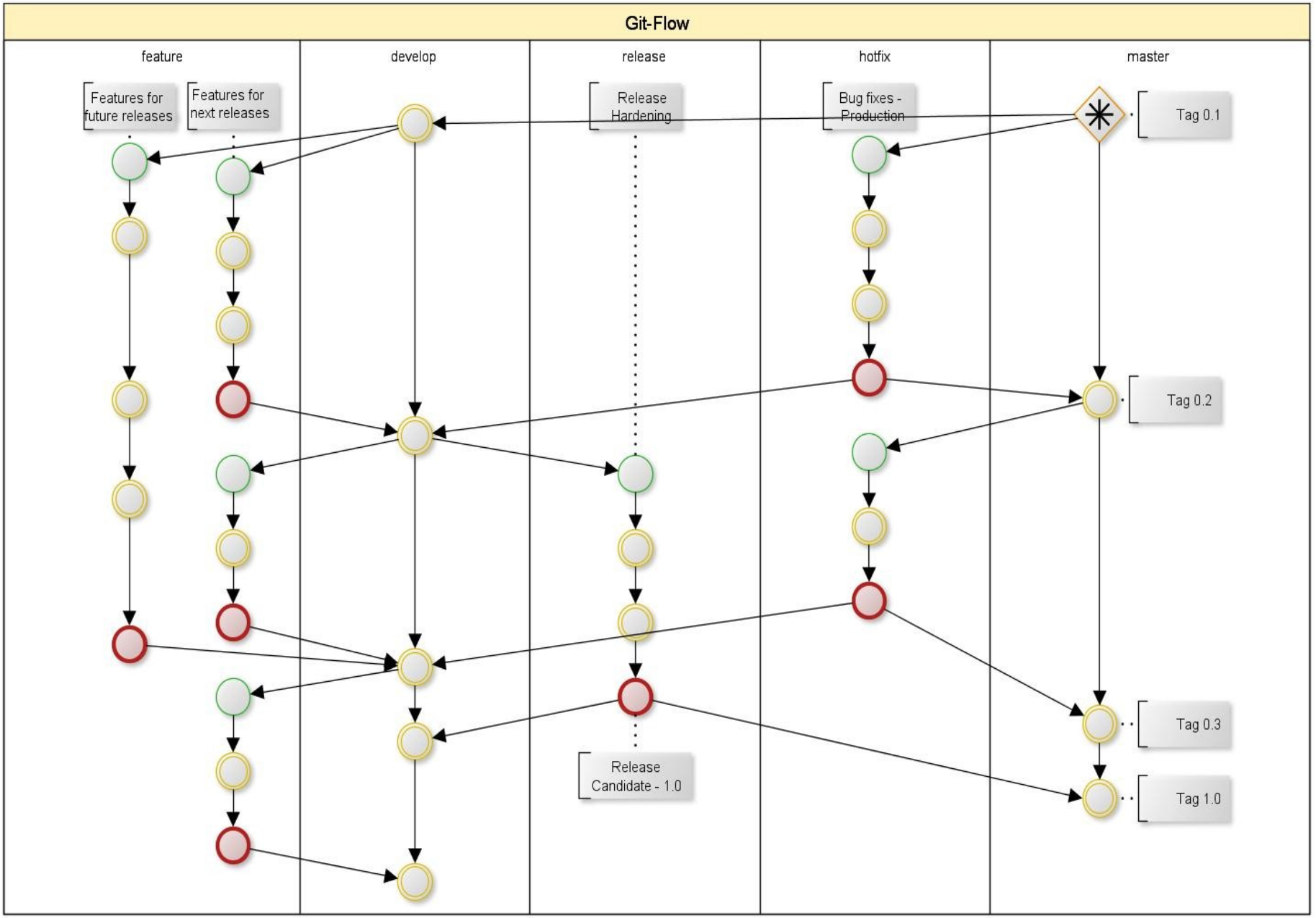
1 branche à durée de vie longue

- `release` → Maturation de la release candidate

X branches à durée de vie (très) courte

- Développement de features (1 branche/feature)
- Bug fixes (1 branche/bug)

git-flow


Git-Flow
feature
develop
release
hotfix
master
Features for future releases
Features for next releases
Release Hardening
Bug fixes - Production
Tag 0.1
Tag 0.2
Tag 0.3
Tag 1.0
Release Candidate - 1.0