

git vs. SVN

Décentralisé

Pas de dépendance au réseau

1+ dépôt local/utilisateur (clone d'un dépôt distant)

1 étape supplémentaire

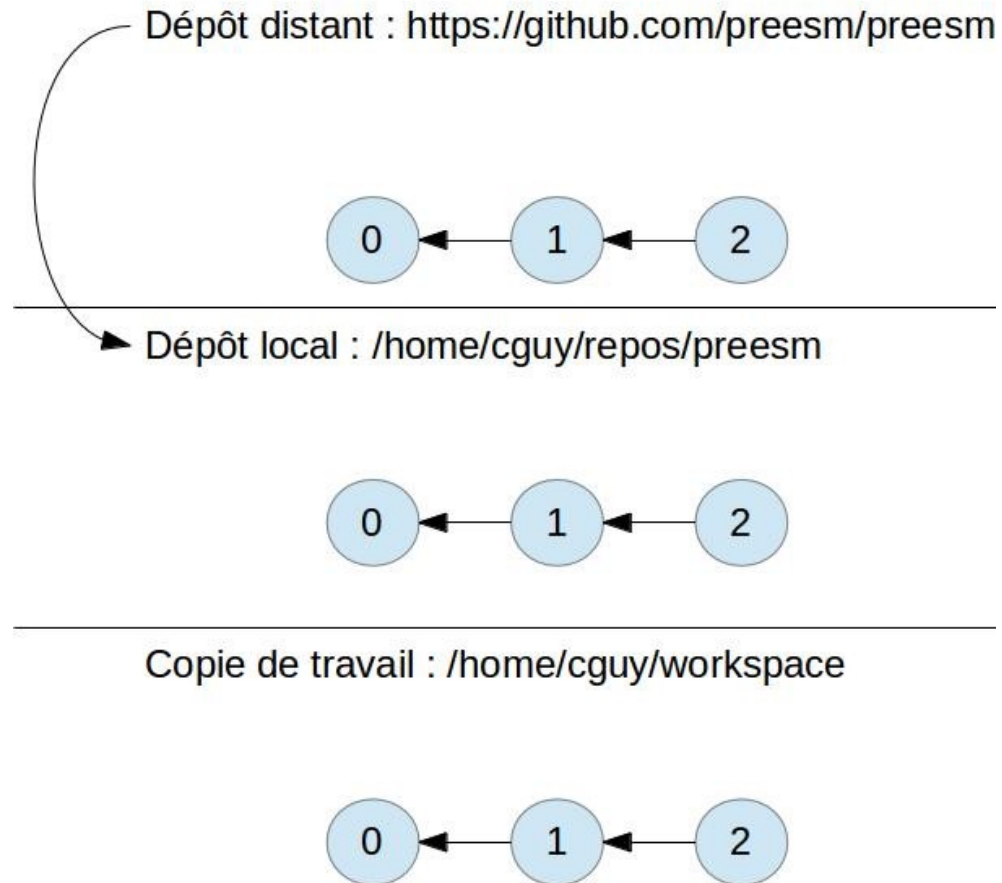
commit (local) + push (distant)

Gestion des branches

Tags

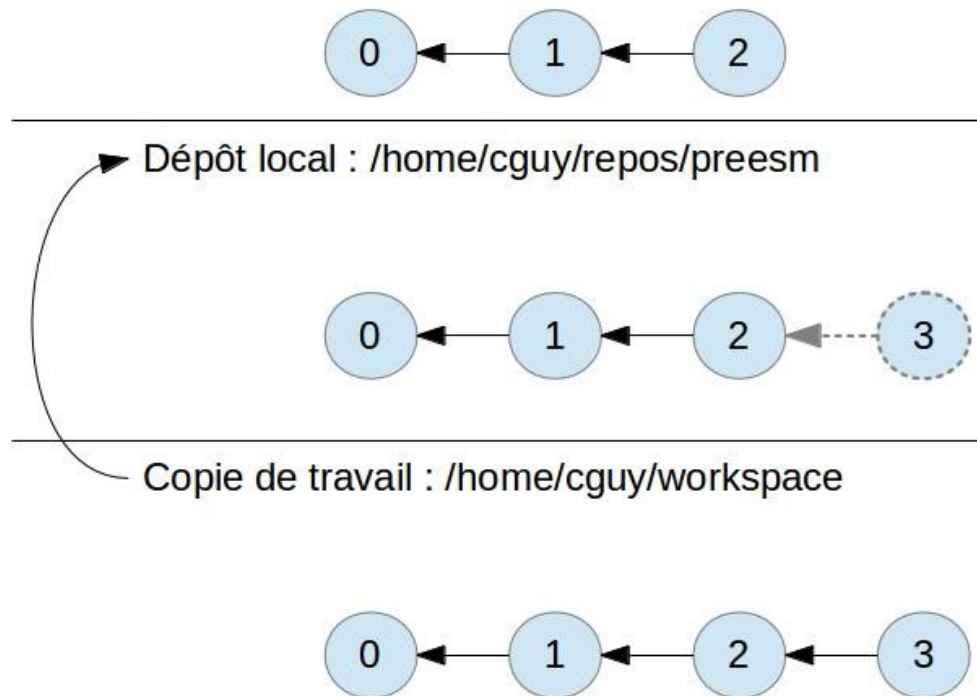
Légère, simple, rapide

Clone

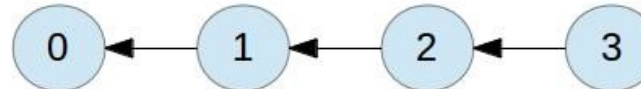
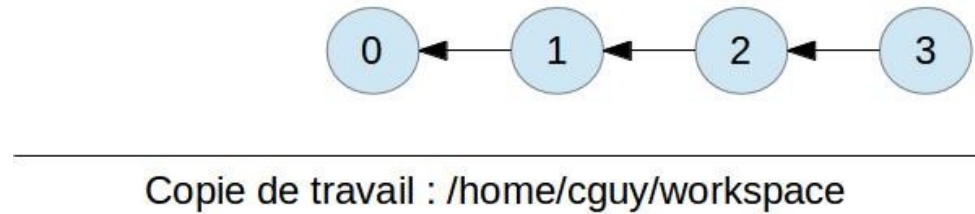
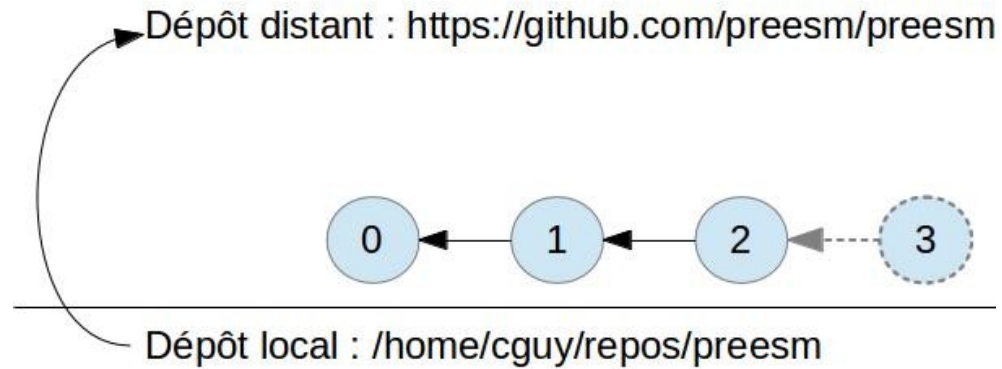


Commit

Dépôt distant : <https://github.com/preesm/preesm>



Push



Mise en place

- Installer git & git-gui
 - Windows
 - Télécharger l'installeur : <http://msysgit.github.com/>
 - Ubuntu
 - `apt-get install git`
 - `apt-get install git-gui`
 - Mac
 - Télécharger l'installeur : <http://code.google.com/p/git-osx-installer>
- Configurer git
 - `git config --global user.name <name>`
 - `git config -global user.email <mail>`
- Créer un compte github
 - Mêmes username et email

Termes usuels

master

Branche principale

HEAD

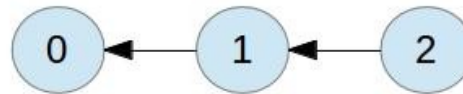
Pointeur sur le dernier commit de la branche courante

origin

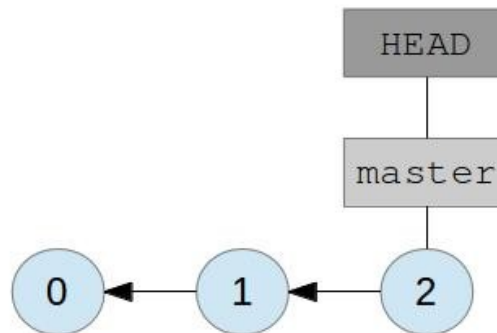
Dépôt distant qui a été cloné pour créer le dépôt local

Termes usuels

Dépôt distant : <https://github.com/preesm/preesm> — `origin`



Dépôt local : `/home/cguy/repos/preesm`



Commandes Git

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

Création de dépôt

```
git init <repo>
```

Initialise un dépôt git local (.git)

```
git clone <url>
```

Crée 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

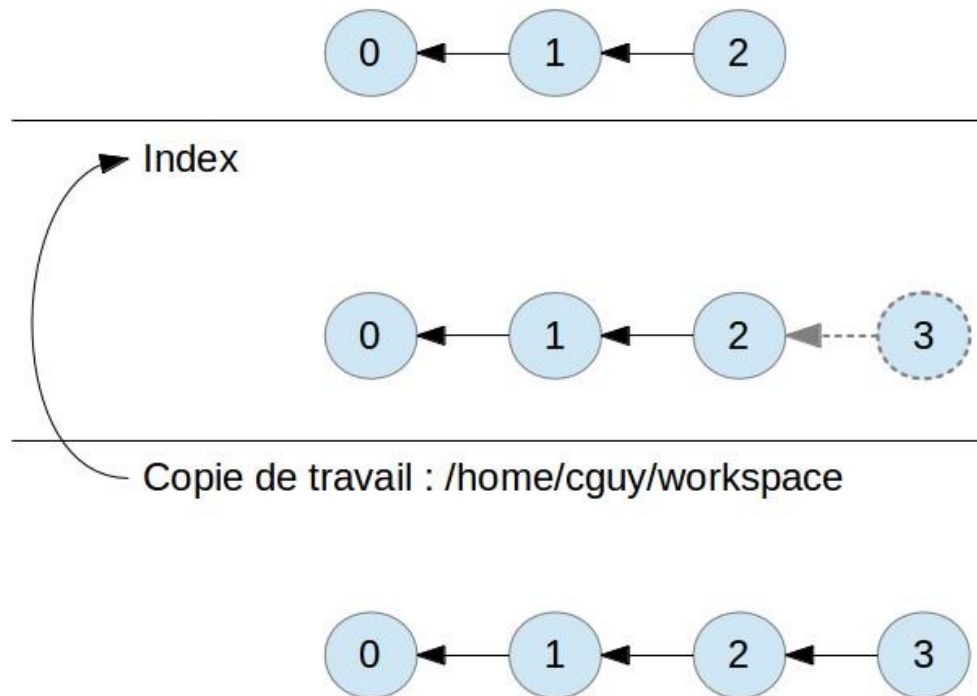
`--staged` supprime un fichier de l'index

```
git mv
```

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

git add

Dépôt local : /home/cguy/repos/preesm



Gestion des commits

```
git commit [-m <message>]
```

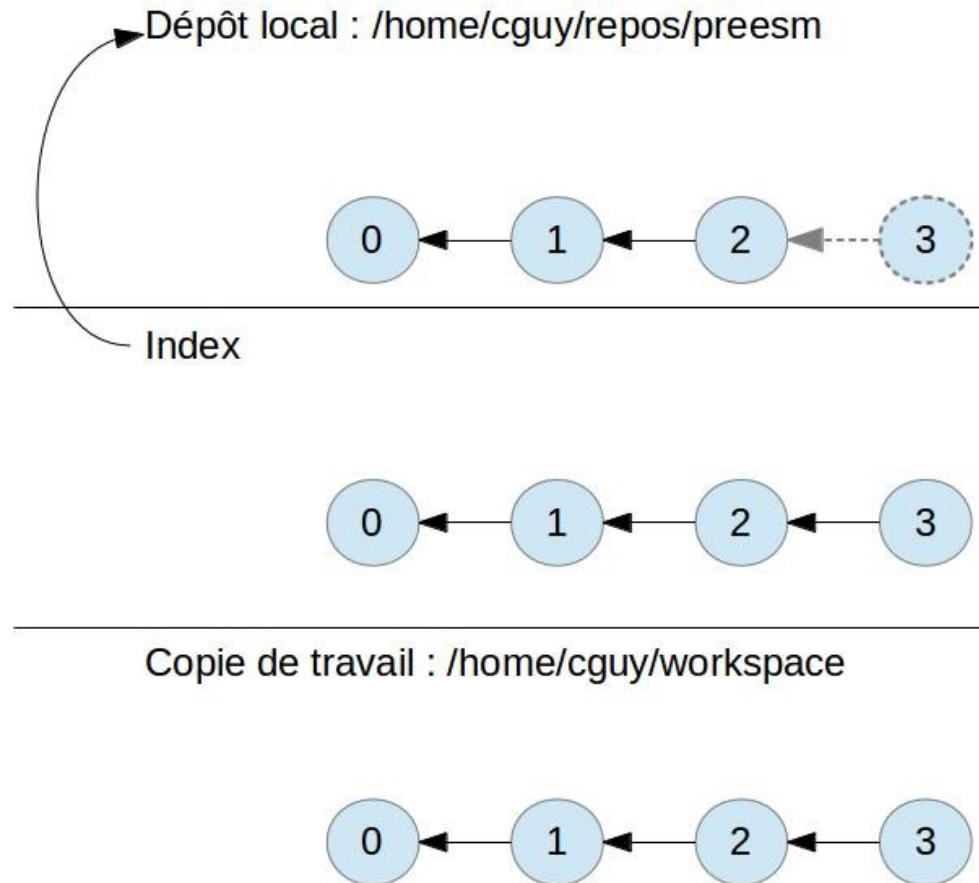
Committe les modifications indexées dans le dépôt **local**

```
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



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 `HEAD`
- 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

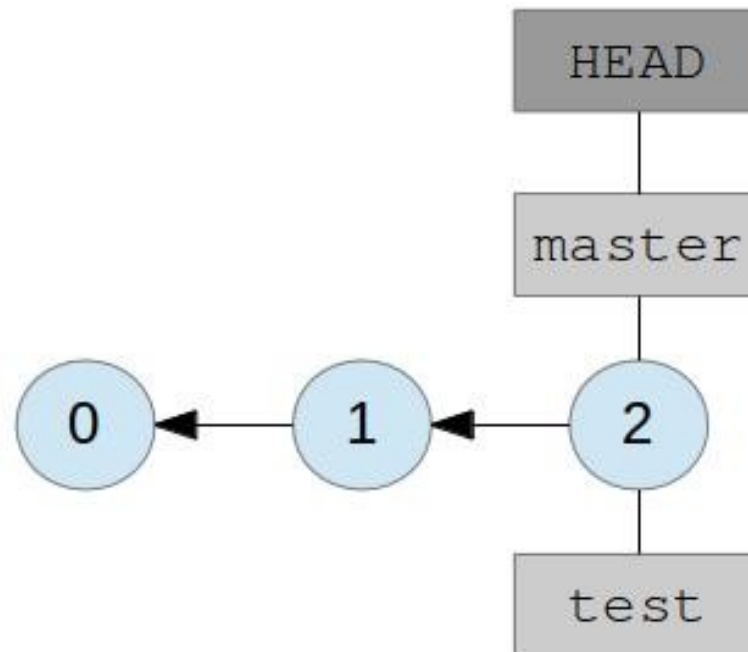
`-b` crée la branche branch avant de basculer dessus

`git stash`

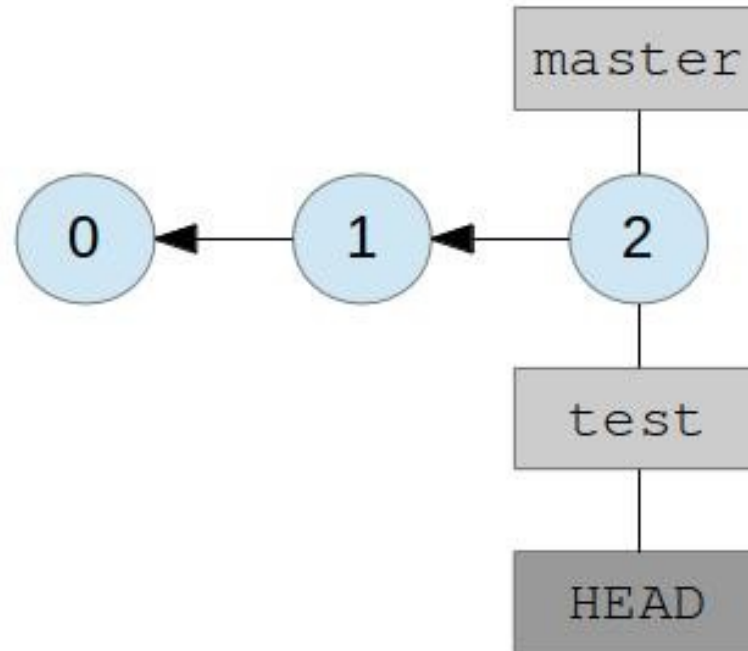
Remise les modifications non-committés

`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

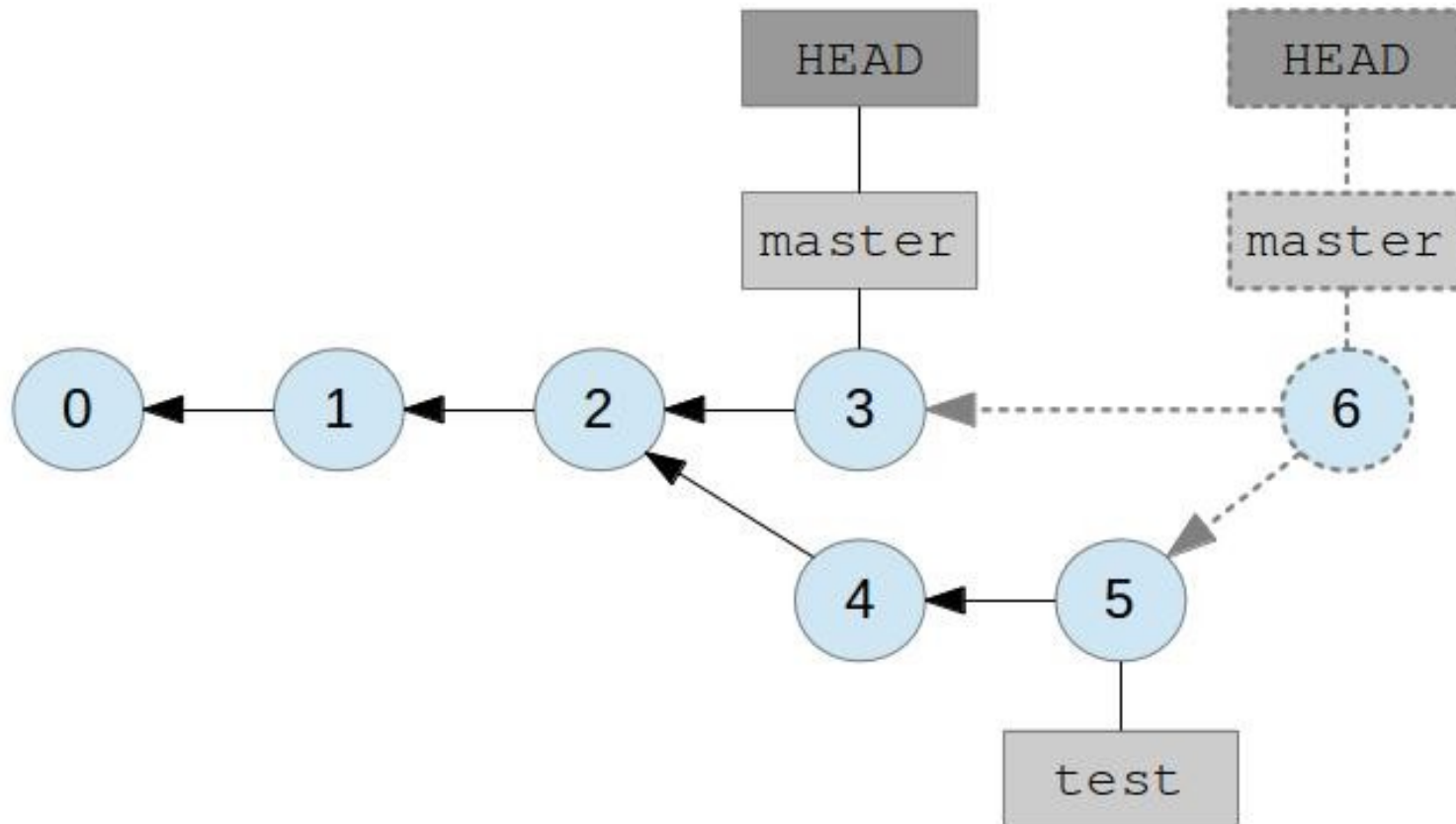
En cas de conflit, le commit est interrompu jusqu'à la résolution du conflit

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 branch -u <repo>/<branch>
```

La branche courante suit la branche distante `branch`

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

Crée une branche locale `localbranch` **qui 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

