3. GIT advanced CLI

Devenez Git Master

Jour 2 / Matin et début d'après-midi

- Matin : Les commandes avancées de GIT (TP)
  1. Plomberie et porcelaine
  2. Développer, Inspecter, Partager, Fusionner, Patcher, Déboguer, Administrer
  3. TP

- Après-midi
  3. Submodules vs Subtree
  4. Hooks et Gitattributes
  5. QR
  6. Revue de code

1. Plomberie et porcelaine

GIT dispose de tout un ensemble d’actions pour les tâches **bas niveau** qui étaient conçues pour être liées dans le style UNIX ou appelées depuis des scripts.

Ces commandes sont dites commandes de « **plomberie** » (*plumbing*) et les autres, plus conviviales sont appelées « la porcelaine » (*porcelain*) - toutes celles que nous avons vu jusqu'à maintenant.

1. Plomberie

- ls-tree
- cat-file = read git file info
  - -t (type)
  - -s (size)
  - -p (pretty print)
- ls-files = read working copy files
  - -c, --cached
  - -d, --deleted
  - -m, --modified
  - -o, --others
  - -i, --ignored
  - -s, --stage
  - ...

- for-each-ref

- hash-object = créer le git SHA-1

```
~$ echo "contenu d'un fichier" | git hash-object --stdin
 c35704bfd99b00045850cb15a51fbae2912d3e43
```

1. Plomberie

git_wc

```
~$ echo "contenu d'un fichier" | git hash-object --stdin
 c35704bfd99b00045850cb15a51fbae2912d3e43

~$ echo "contenu d'un fichier" | git hash-object --stdin -w
~$ ls .git/objects/c3/704bfd99b00045850cb15a51fbae2912d3e43

~$ git cat-file -p c35704bfd
contenu d'un fichier
```

1. Plomberie

git_wc

```
~$ git cat-file -p HEAD
tree 0450b171baef843496abb95eeed6e2b705a97c95
parent c85737aad7cddb032125e7d07749961f516b14cd
author Sylvain Just <sy.just@i-sloth.com> 1461510514 +0200
committer Sylvain Just <sy.just@i-sloth.com> 1461510520 +0200

readme figlet <pre> tags

~$ git ls-tree -r HEAD
100755 blob d0d7ade2f4a307172068673a8739952bae37326d    bin/script.sh
100644 blob 277e22cab111190194ef8701296ed380650f195a    lib/script.inc.sh
100644 blob 36ad7da7593920668d299764778996b189037883    readme.md
```

1. Plomberie

git_wc

à partir des objets dans votre ./git

Trouvez le SHA-1

- d'un commit
- d'un blob
- d'un tree

avec ls-files : affichez au moins un fichier pour chacune des options suivantes

-c, --cached, -d, --deleted, -m, --modified, -o, --others, -i, --ignored, -s, --stage

2. Les commandes avancées de GIT

Travaux Pratiques

- Développer
- Inspecter
- Partager
- Fusionner
- Patcher
- Déboguer
- Administrer

2. Les commandes avancées de GIT

Administrer

- gc --agressive (tout les 100 commit - en cas de lenteur ie)
- fsck - Verifies the connectivity and validity of the objects in the database

git_wc

```
~$ git fsck
Checking object directories: 100% (256/256), done.
Checking objects: 100% (46/46), done.
dangling blob c35704bfd99b00045850cb15a51fbae2912d3e43
```

2. Les commandes avancées de GIT

Administrer

- reflog = historique de HEAD

git_wc

```
~$ git fsck
Checking object directories: 100% (256/256), done.
Checking objects: 100% (46/46), done.
dangling blob c35704bfd99b00045850cb15a51fbae2912d3e43
```

2. Les commandes avancées de GIT

Administrer

- filter-branch
  - --tree-filter <cmd>
  - --msg-filter <cmd>
  - --commit-filter <cmd>
  - ...
  - travaille en local... push --force pour publier sur le dépôt.

```
[limiter l'historique directement au niveau du dépôt]
~$ git filter-branch --tree-filter 'rm -rf /path/to/asset/folder' HEAD
~$ git push --force
```

C'est généralement une bonne idée de le faire dans un branche de test puis de faire une réinitialisation forte (**hard-reset**) de votre branche master

2. Les commandes avancées de GIT

Administrer

```
~$ git filter-branch --tree-filter 'rm -f passwords.txt' HEAD (--all branches)
   Rewrite 6b9b3cf04e7c5686a9cb838c3f36a8cb6a0fc2bd (21/21)
   Ref 'refs/heads/master' was rewritten

~$ git filter-branch --commit-filter '
        if [ "$GIT_AUTHOR_EMAIL" = "schacon@localhost" ];
        then
                GIT_AUTHOR_NAME="Scott Chacon";
                GIT_AUTHOR_EMAIL="schacon@example.com";
                git commit-tree "$@";
        else
                git commit-tree "$@";
        fi' HEAD
```

2. Les commandes avancées de GIT

Administrer

- instaweb
- archive

```
[limiter l'historique directement au niveau du dépôt]
~$ git filter-branch --tree-filter 'rm -rf /path/to/asset/folder' HEAD
```

2. Les commandes avancées de GIT

Développer

reset	au niveau d'un fichier / d'un commit
add	Indexer
commit	Valider
rm, mv, status	...

2. Les commandes avancées de GIT

Développer

- reset [<commit>] file

git-pro reset demystifié

- reset (*option*) <commit>
  - --soft
  - --hard
  - --mixed

2. Les commandes avancées de GIT

Développer

--patch et --interactive

- add --edit (diff file)
- add --patch (interactive edit hunks)
- add --interactive
  (un peu fastidieux à l'usage)

```
~$ git add --edit <file>
~$ git add --patch <file>
~$ git add --interactive <file>


~$ git commit --edit <file>
~$ git commit --patch <file>
~$ git commit --interactive <file>


~$ git reset --patch <file>
```

2. Les commandes avancées de GIT

Développer

--patch et --interactive

Utilisez git add --patch et reset --patch

sur un fichier comprenant de nombreuses lignes et de nombreuses modifications

faites des commits atomiques précis sur certaines lignes uniquement de ce fichier avec des sujets concis traitant exactement de ce qui a été commité

```
~$ git add --edit <file>
~$ git add --patch <file>
~$ git add --interactive <file>


~$ git commit --edit <file>
~$ git commit --patch <file>
~$ git commit --interactive <file>


~$ git reset --patch <file>
```

2. Les commandes avancées de GIT

Développer

- commit --amend

Modifiez le dernier commit de votre exercie précédent

```
~$ git lg -1 HEAD
   7f77b4f bfile mod 2 txt

   … some modifications
~$ git add new-file.txt
~$ git commit --amend
   …

~$ git lg -1 HEAD
   f6269bb bfile mod 2 text
```

2. Les commandes avancées de GIT

Inspecter / Comparer

sjtp.mc-remote

- log
- show
- shortlog
- diff / difftool
- describe

```
* aa017d9d - (HEAD, master) readme figlet
*-.   c85737aa - (tag: v0.0.2, syl) Merge
|\ \
| | * d11aefb1 - readme mod
| * | ad856f54 - (john) add comment on ma
| * | 217c8fd2 - include version
| * | 747f5781 - add args in error messag
| |/
* | 97b2ef57 - (jane) jane script lib aut
|/
*   95f1fc4d - (tag: v0.0.1) Merge remote
|\
| * 592f8cd2 - myFunc return bug fix
* | ef3fabc4 - jane func implementation
|/
* 01529ed0 - first commit (2016-04-18 15:
```

2. Les commandes avancées de GIT

Inspecter / Comparer

- show

```
--pretty[=<format>], --format=<format>
--oneline
--abbrev-commit
--no-abbrev-commit
--oneline
```

décrit les 3 types d'objets avec un message et un diff

- blob
- tree
- commit

mix entre log et les commandes de plomberie

- cat-file
- ls-tree
- diff-tree

2. Les commandes avancées de GIT

Inspecter / Comparer

sjtp.mc-remote

- log
- git log HEAD^
- git log c8573^2 (2ème parents - *merge*)
- git log c8573~3 (git log c8573^^^)

git_wc

- git log -Smc\.com

```
* c0501ecb - (HEAD, master, iss53)
...
* bbe79bd3 - email here
```

```
* aa017d9d - (HEAD, master) readme figlet
*-.   c85737aa - (tag: v0.0.2, syl) Merge
|\ \
| | * d11aefb1 - readme mod
| * | ad856f54 - (john) add comment on ma
| * | 217c8fd2 - include version
| * | 747f5781 - add args in error messag
| |/
* | 97b2ef57 - (jane) jane script lib aut
|/
*   95f1fc4d - (tag: v0.0.1) Merge remote
|\
| * 592f8cd2 - myFunc return bug fix
* | ef3fabc4 - jane func implementation
|/
* 01529ed0 - first commit (2016-04-18 15:
```

2. Les commandes avancées de GIT

Inspecter / Comparer

sjtp.mc-remote

```
* aa017d9d - (HEAD, master) readme figlet
*-.   c85737aa - (tag: v0.0.2, syl) Merge
|\ \
| | * d11aefb1 - readme mod
| * | ad856f54 - (john) add comment on ma
| * | 217c8fd2 - include version
| * | 747f5781 - add args in error messag
| |/
* | 97b2ef57 - (jane) jane script lib aut
|/
*   95f1fc4d - (tag: v0.0.1) Merge remote
|\
| * 592f8cd2 - myFunc return bug fix
* | ef3fabc4 - jane func implementation
|/
* 01529ed0 - first commit (2016-04-18 15:
```

- git log john..jane (in jane only)
  - = git log jane --not john

```
97b2ef57 - (origin/jane, jane)
```

- git log --left-right john...jane

```
< ad856f54 - (origin/john, john)
< 217c8fd2 - include version
< 747f5781 - add args in error
> 97b2ef57 - (origin/jane, jane)
```

2. Les commandes avancées de GIT

Inspecter / Comparer

sjtp.mc-remote

- shortlog

```
Jane Bar (2):
      jane func implementation
      jane script lib author

John Foo (6):
      first commit
      myFunc return bug fix (one arg bad return)
      add args in error message
      include version
      add comment on main function
      Merge 3 remote-tracking branches into integration

Sylvain Just (2):
      Merge remote-tracking branch 'origin/john' into integration
      readme mod
```

2. Les commandes avancées de GIT

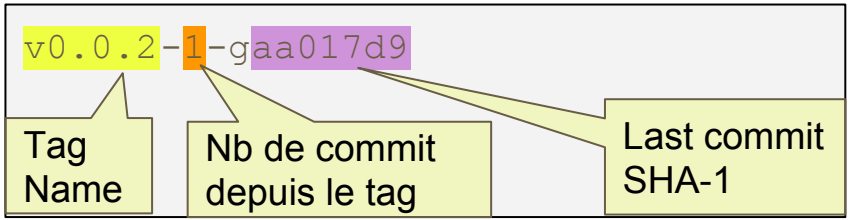
Inspecter / Comparer

- diff / difftool

```
~$ git diff readme.md
--- a/readme.md
+++ b/readme.md
@@ -6,3 +6,4 @@
+# a modif
```

- Voir les modification à entre les fichiers indexés et ceux de la copie de travail (Modified)...
- **Difftool** permet d'utiliser l'éditeur pour voir les modification
- **diff --check** *(eol whitespaces warnings)*
- voir le dernier tag sur de la branche courante
  (ou de toutes les branches avec *--all)*

- describe



2. Les commandes avancées de GIT

Fusionner

- merge --no-commit / --squash

```
git config --global \
merge= "merge  --no-commit"
```

- mergetool (editeur visuel)

- stash (mettre son travail de côté)

2. Les commandes avancées de GIT

stash & clean

- mettre de côté / empiler

```
~$ git status
   M afile.txt
   ?? bfile.txt
~$ git stash
~$ git stash -u
~$ git stash list
 stash@{0}: WIP on master: 7cb1915
(A)
 stash@{1}: WIP on master: 7cb1915
(M)
```

- récupérer / dépiler

```
~$ git stash apply stash@{1}
~$ git stash pop
~$ git stash list
    stash@{0}: WIP ...7cb1915 (A)

~$ git status
   M afile.txt
   ?? bfile.txt
```

- nettoyer

```
~$ git clean -n -d (dry run / untracked-(empty)-dir)
~$ git clean -f -i (force / interactive)
~$ git clean -x -e (ignore .gitignore / take regex)
~$ git clean -X (only .gitignore patterns)
```

2. Les commandes avancées de GIT

Patcher

• apply • format-patch / am (mail)	générer et appliquer des patch
• cherry-pick	merger de façon chirurgicale d'une branche à l'autre (seulement quelques commits)
• rebase • revert	modifier son historique

2. Les commandes avancées de GIT

Patcher

sjtp.mc-remote

- apply
- format-patch / am (mail)

générer et appliquer des patch

```
~$ git checkout john
```

2. Les commandes avancées de GIT

Patcher

sjtp.mc-remote

- apply
- format-patch / am (mail)

```
~$ git checkout john
```

```
* aa017d9d - (origin/master, origin/HEAD, master)
*-.   c85737aa - (tag: v0.0.2, origin/syl) Merge
|\ \
| | * d11aefb1 - readme mod (8 days ago) <Sylvain
| * | ad856f54 - (HEAD, origin/john, john) add co
| * | 217c8fd2 - include version (8 days ago) <Jo
| * | 747f5781 - add args in error message (8 day
| |/
* | 97b2ef57 - (tag: v0.0.1b, origin/jane, jane)
|/
*   95f1fc4d - (tag: v0.0.1) Merge remote-trackin
|\
| * 592f8cd2 - myFunc return bug fix (one arg bad
* | ef3fabc4 - jane func implementation (9 days a
|/
* 01529ed0 - (origin/first_commit_branch) first c
```

2. Les commandes avancées de GIT

Patcher

sjtp.mc-remote

- apply
- format-patch / am (mail)

générer et appliquer des patch

```
~$ git checkout john
~$ git format-patch HEAD~3

   0001-add-args-in-error-message.patch
   0002-include-version.patch
   0003-add-comment-on-main-function.patch
```

2. Les commandes avancées de GIT

Patcher

sjtp.mc-remote

- apply — générer et appliquer des patch
- format-patch / am (mail)

```
~$ cat   0001-add-args-in-error-message.patch

From 747f5781df24012be1e98559c207f97cbb05e0a9 Mon Sep 17 00:00:00 2001
From: John Foo <j.foo@i-sloth.com>
Date: Mon, 18 Apr 2016 17:24:32 +0200
Subject: [PATCH 1/3] add args in error message

---
```

2. Les commandes avancées de GIT

Patcher

sjtp.mc-remote

- apply
- format-patch / am (mail)

```
~$ git checkout -b br-v1 v0.0.1
```

```
* aa017d9d - (origin/master, origin/HEAD, master)
*-.   c85737aa - (tag: v0.0.2, origin/syl) Merge
|\ \
| | * d11aefb1 - readme mod (8 days ago) <Sylvain
| * | ad856f54 - (HEAD, origin/john, john) add co
| * | 217c8fd2 - include version (8 days ago) <Jo
| * | 747f5781 - add args in error message (8 day
| |/
* | 97b2ef57 - (tag: v0.0.1b, origin/jane, jane)
|/
*   95f1fc4d - (tag: v0.0.1) Merge remote-trackin
|\
| * 592f8cd2 - myFunc return bug fix (one arg bad
* | ef3fabc4 - jane func implementation (9 days a
|/
* 01529ed0 - (origin/first_commit_branch) first c
```

2. Les commandes avancées de GIT

Patcher

sjtp.mc-remote

- apply
- format-patch / am (mail)

générer et appliquer des patch

```
~$ git apply *.patch
~$ git diff
diff --git a/bin/script.sh b/bin/script.sh
index 3988ff5..d0d7ade 100755
--- a/bin/script.sh
+++ b/bin/script.sh
...
~$ git status -bs
## br-v1
 M bin/script.sh
```

2. Les commandes avancées de GIT

Patcher

sjtp.mc-remote

- apply
- format-patch / am (mail)

générer et appliquer des patch

```
~$ git checkout bin
~$ git am *patch
Applying: add args in error message
Applying: include version
Applying: add comment on main function

[on error]
git am --continue
git am --skip
git am --abort
```

```
* c9d9795d - (HEAD, br-v1)
   add comment on main...
* 477164e5 - include version
* 5312c4f6 - add args in err
*   95f1fc4d - (tag: v0.0.1)
|\
| * 592f8cd2 - myFunc return
* | ef3fabc4 - jane func imp
|/
* 01529ed0 - (origin/first_c
```

2. Les commandes avancées de GIT

Patcher

sjtp.mc-remote

```
* aa017d9d - (origin/master, origin/HEAD, master)
*-.   c85737aa - (tag: v0.0.2, origin/syl) Merge
|\ \
| | * d11aefb1 - readme mod (8 days ago) <Sylvain
| * | ad856f54 - (HEAD, origin/john, john) add co
| * | 217c8fd2 - include version (8 days ago) <Jo
| * | 747f5781 - add args in error message (8 day
| |/
* | 97b2ef57 - (tag: v0.0.1b, origin/jane, jane)
|/
*   95f1fc4d - (tag: v0.0.1) Merge remote-trackin
|\
| * 592f8cd2 - myFunc return bug fix (one arg bad
* | ef3fabc4 - jane func implementation (9 days a
|/
* 01529ed0 - (origin/first_commit_branch) first c
```

générer et appliquer des patch

```
* c9d9795d - (HEAD, br-v1)
    add comment on main...
* 477164e5 - include version
* 5312c4f6 - add args in err
*   95f1fc4d - (tag: v0.0.1)
|\
| * 592f8cd2 - myFunc return
* | ef3fabc4 - jane func imp
|/
* 01529ed0 - (origin/first_c
```

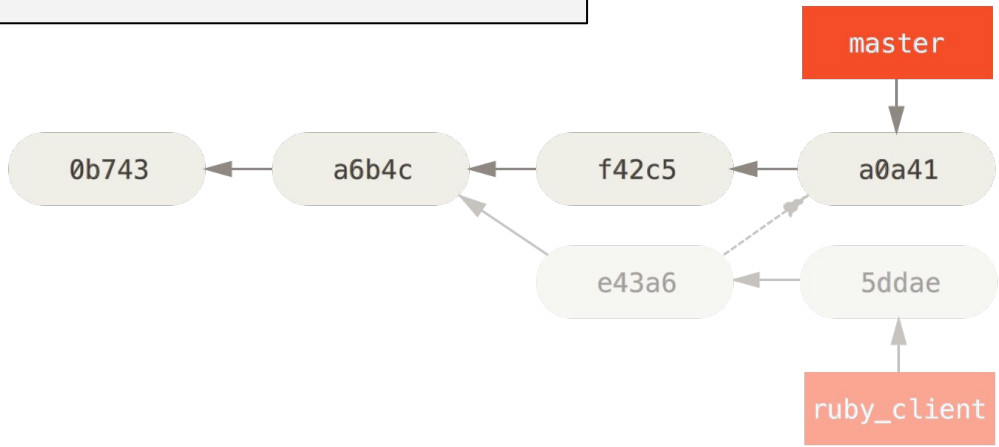
2. Les commandes avancées de GIT

Patcher (cherry-pick)

- cherry-pick

merger de façon chirurgicale d'une branche à l'autre (seulement quelques commits)

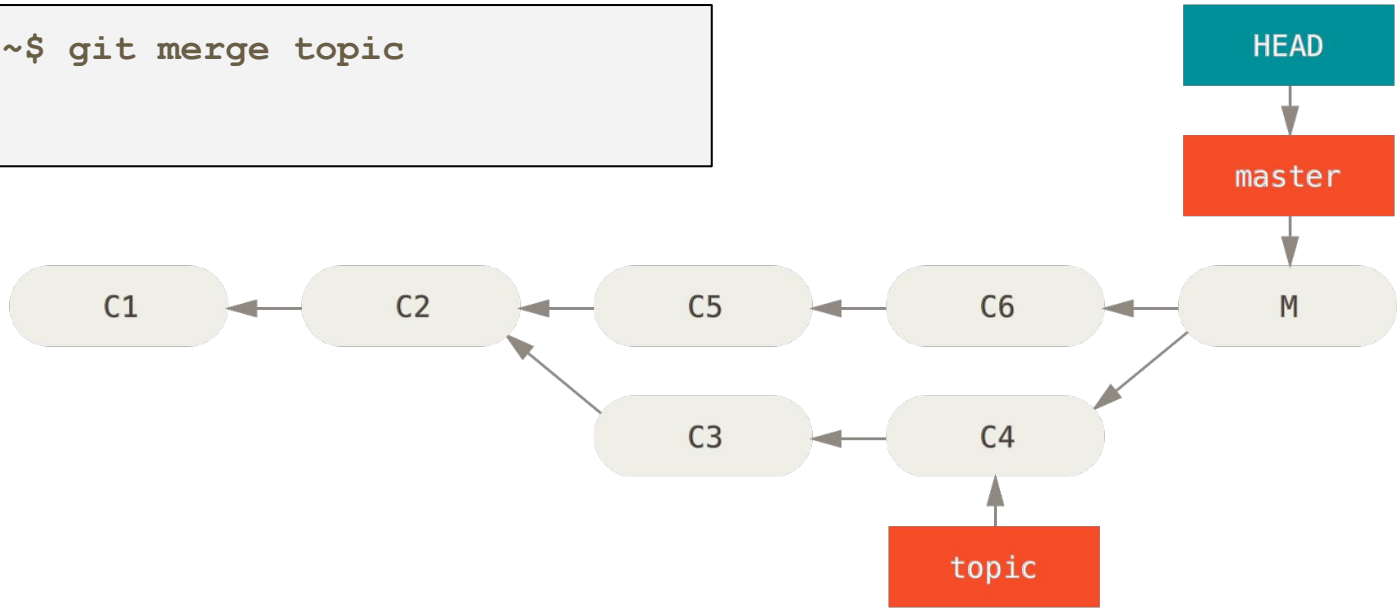
```
~$ git checkout master
~$ git cherry-pick e43a6
```



2. Les commandes avancées de GIT

Patcher (revert)

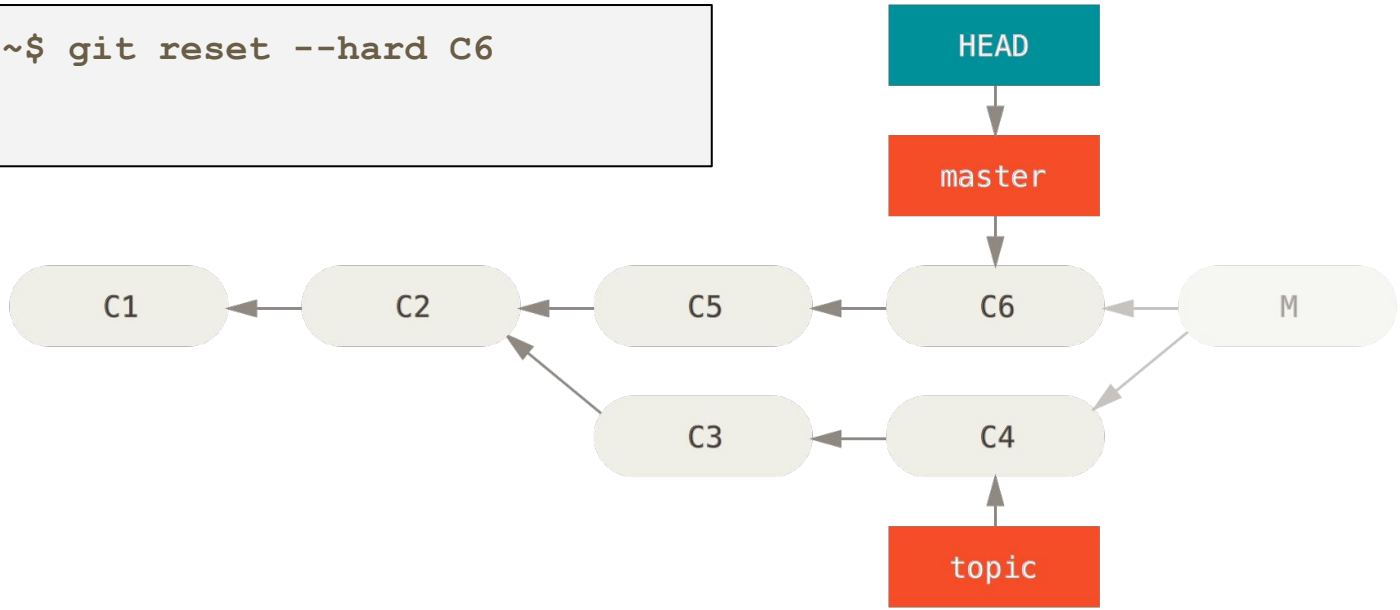
```
~$ git merge topic
```



2. Les commandes avancées de GIT

Patcher (revert)

```
~$ git reset --hard C6
```

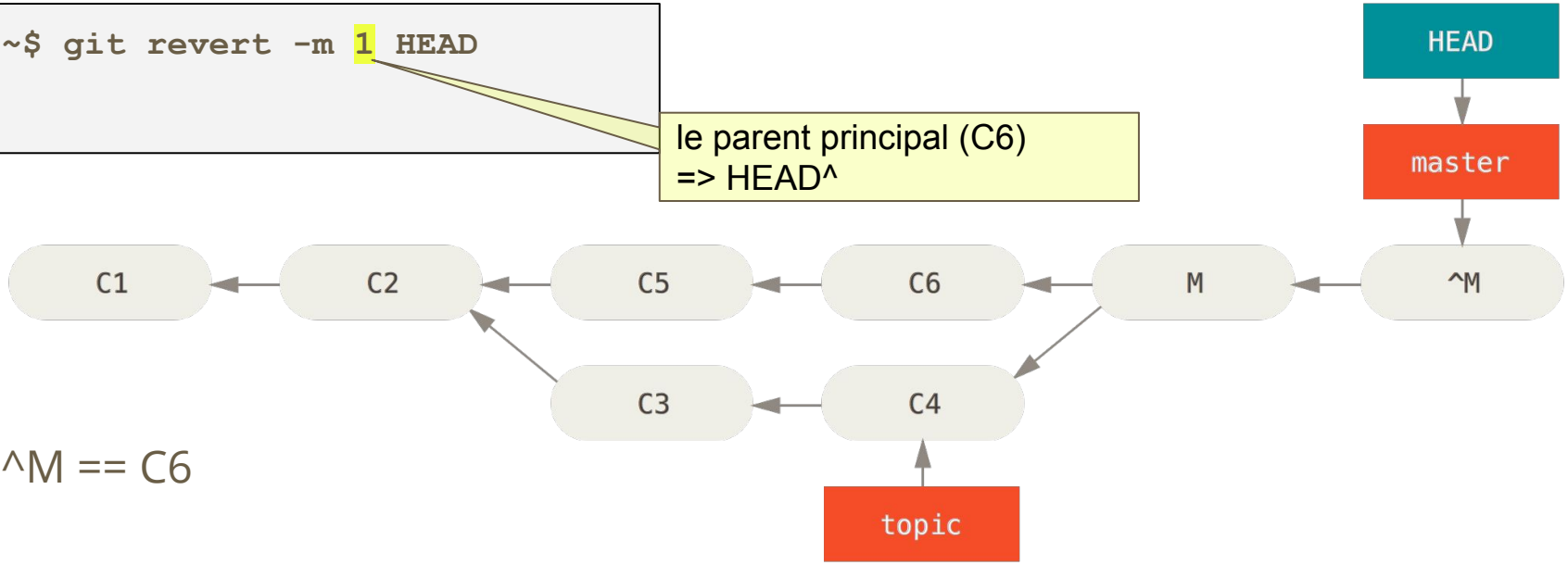


2. Les commandes avancées de GIT

Patcher (revert)

```
~$ git revert -m 1 HEAD
```

le parent principal (C6)
=> HEAD^

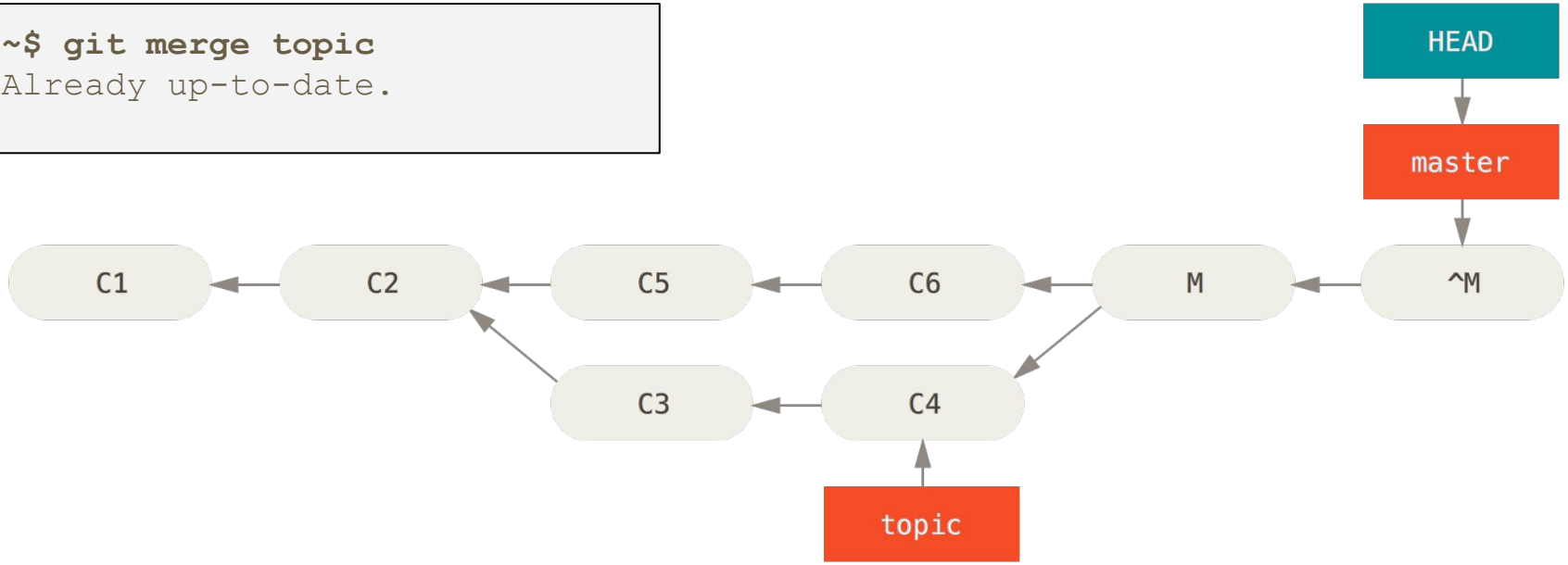


^M == C6

2. Les commandes avancées de GIT

Patcher (revert)

```
~$ git merge topic
Already up-to-date.
```



2. Les commandes avancées de GIT

Patcher (rebase -i)

```
~$ git rebase -i HEAD~4
     pick 19190d1 afile mod
     pick 4ce694e added bfile.txt
     pick f131f32 bfile mod 1
     pick f6269bb bfile mod 2

# Rebase 7cb1915..aba07fe onto 7cb1915
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#  x, exec = run command (the rest of the line) using shell
```

2. Les commandes avancées de GIT

Patcher (rebase -i)

```
~$ git rebase -i HEAD~4
     pick 19190d1 afile mod
     pick 4ce694e added bfile.txt
     pick f131f32 bfile mod 1
     pick f6269bb bfile mod 2

# Rebase 7cb1915..aba07fe onto 7cb1915
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#  x, exec = run command (the rest of the line) using shell
```

```
reword 4ce694e added bfile.txt
fixup f131f32 bfile mod 1
squash f6269bb bfile mod 2 text
pick 19190d1 afile mod
```

2. Les commandes avancées de GIT

Patcher (rebase -i)

```
# This is a combination of 3 commits.
# The first commit's message is:
added bfile.txt with many lines

# The 2nd commit message will be skipped:

# bfile mod 1

# This is the 3rd commit message:

bfile mod 2 text
```

```
reword 4ce694e added bfile.txt
fixup f131f32 bfile mod 1
squash f6269bb bfile mod 2 text
pick 19190d1 afile mod
```

```
~$ git log --abbrev-commit
9807034 afile mod
17ba8ce added bfile.txt with ...
```

2. Les commandes avancées de GIT

Déboguer

- blame <file>
- grep
- log -S

- bisect

outil à workflow (start | bad | good)
dernier commit bad connu
dernier commit good connu
bisect fait un checkout à chaque étape du chemin et nous demande de valider par good ou bad

2. Les commandes avancées de GIT

TP

réunissez vous par équipe
créer un dépôt avec des fichiers de code que vous connaissez (depuis un export svn ?)
ignorez les fichiers que vous ne voulez pas commiter
créez une branche de développement
créez une branche par personne (dévelopeur)
créez un dépôt distant par équipe (lecture écriture pour les membres de l'équipe seulement)
gérez un conflit en mergeant sur une branche de feature (ou dévelopeur)
gérez un rebase sur la branche de dev
gérer un ou plusieurs merge en fast forward de dev sur master
créez 2 tags au moins
ajoutez des attributs publics pour voir un diff humainement lisible sur les images de votre dépôt
ajoutez un hook sur le serveur pour empêcher les commit avec la chaine de caractère "toto"
imprimez le graph de votre travail et encadrez le

3. Submodules vs Subtree

[submodule in git book] *Submodules*

```
~$ git submodule add https://github.com/syjust/sjtp.mc-init


[charger un projet et tous ces sous modules]
~$ git clone --recursive <project url>
```

3. Submodules vs Subtree

(git-attitude.fr//git-subtrees)

Subtree

```
~$ git remote add mcinit https://github.com/syjust/sjtp.mc-init
~$ git fetch mcinit
~$ git read-tree --prefix=lib/mcinit -u mcinit/master
~$ git status
A  lib/bashLib/LICENCE.md
A  lib/bashLib/README.md
A  lib/bashLib/bin/es.sh
A  lib/bashLib/bin/get-conf.sh
```

4. Hooks & Gitattributes

- Git hooks sur git pro 8.3 et 8.4
  - quels hooks pourriez vous mettre en place ?
  - faites le.

- Git attributes sur git pro 8.2
  - mettre en place un attribut diff utilisant la commande exif
  - mettre en place un attribut filter.date

4. Hooks & Gitattributes

TP : pour finir

réunissez vous par équipe
créer un dépôt avec des fichiers de code que vous connaissez (depuis un export svn ?)
ignorez les fichiers que vous ne voulez pas commiter
créez une branche de développement
créez une branche par personne (développeur)
créez un dépôt distant par équipe (lecture écriture pour les membres de l'équipe seulement)
gérez un conflit en mergeant sur une branche de feature (ou développeur)
gérez un rebase sur la branche de dev
gérer un ou plusieurs merge en fast forward de dev sur master
créez 2 tags au moins
ajoutez des attributs publics pour voir un diff humainement lisible sur les images de votre dépôt
ajoutez un hook sur le serveur pour empêcher les commit avec la chaine de caractère "toto"

5. Questions / Réponses