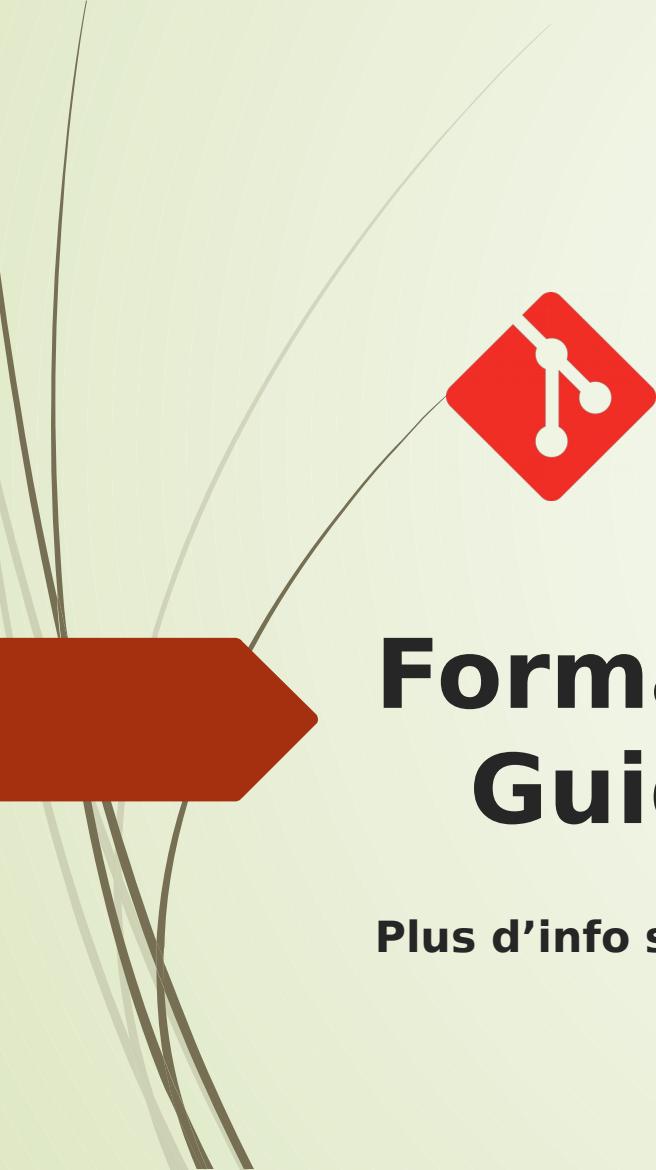


DAWAN Paris
DAWAN Nantes
DAWAN Lyon

11,rue Antoine Bourdelle, 75015 PARIS
32, Bd Vincent Gâche, 5e étage - 44200 NANTES
Bt Banque Rhône Alpes, 2ème étage - 235 cours Lafayette 69006 LYON



Formation Git: Guide pratique

Plus d'info sur <http://www.dawan.fr> ou **0810.001.917**

Formateur: Matthieu LAMAMRA



Présentation de la formation

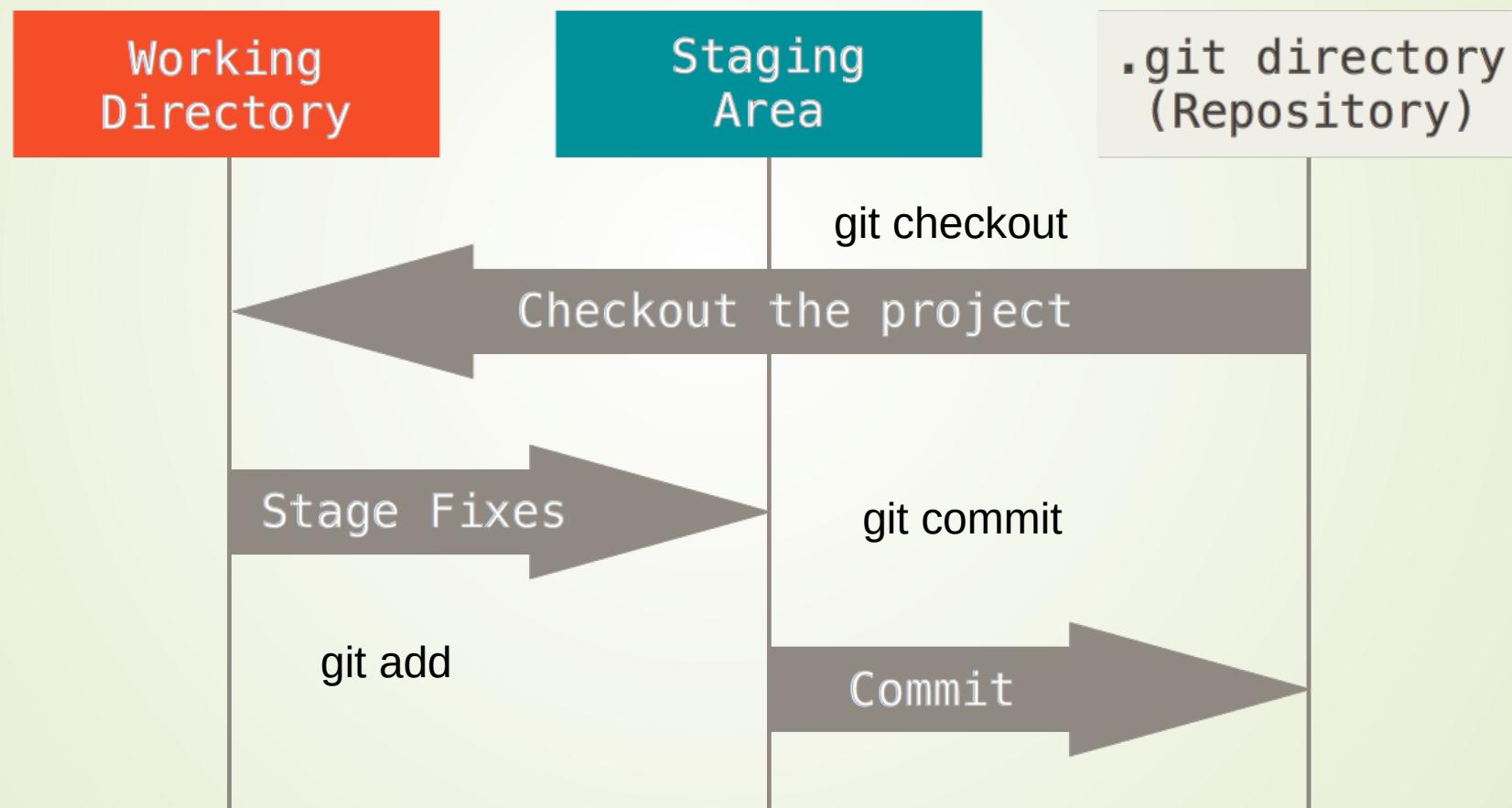
- Rappeler le contexte d'utilisation des commandes
- Montrer les commandes avec leurs options principales
- Montrer les résultats des commandes

Gestion de Code Source

- Définition
 - Logiciel qui permet de stocker un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus. Il permet notamment de retrouver les différentes versions d'un lot de fichiers connexes, nommés « commits ».
 - Les avantages de git :
 - le travail est sauvegardé régulièrement, permettant des rollbacks en cas de perte de données
 - l'outil fournit une série de métadonnées détaillant les commits (date, auteur, détail des modifications..)
 - git est décentralisé, chaque collaborateur possède une version du code dans son dépôt local, et travaille indépendamment des autres

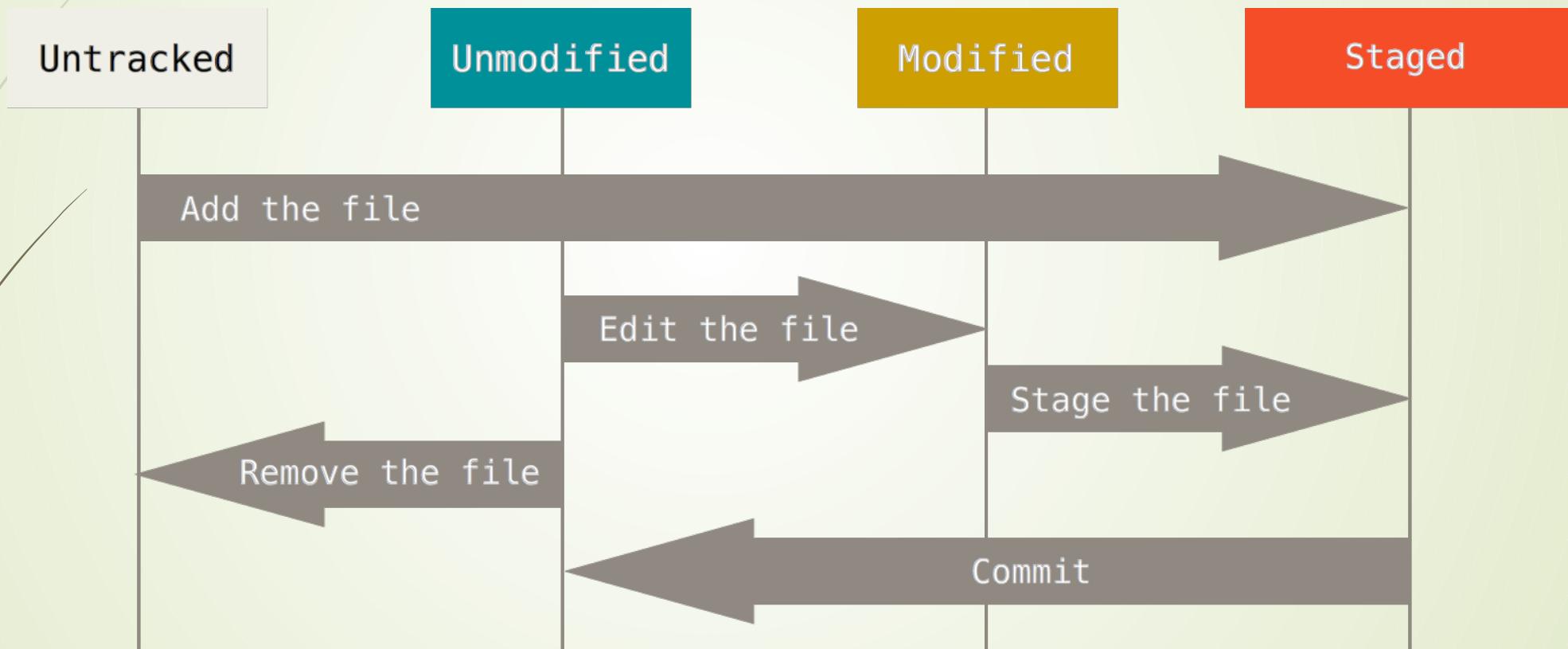
Dépôt local

- Les trois arborescences (zones) du dépôt



Dépôt local

- Les état d'un fichier dans le dépôt



Dépôt local

- Cr ation du d p t
 - **git init [dir_name]**

```
ll depot  
30 12:29 ./  
30 12:29 ../  
30 12:29 .git/
```

Dépôt local

- Edition des métadonnées utilisateur
 - **git config** <metadata> [<value>]

```
git config --local user.name mlamamra  
git config --global user.email mlamamra@dawan.fr
```

.git/config
~/.gitconfig

<https://git-scm.com/docs/git-config>

Dépôt local

- Statut du dépôt

- **git status** [-s | --short]

```
matthieu@matthieu:~/depot$ git status
Sur la branche master

Aucun commit

rien à valider (créez/copiez des fichiers
et utilisez "git add" pour les suivre)
```

<https://git-scm.com/docs/git-status>

Dépôt local

- nouveau fichier non suivi

```
matthieu@matthieu:~/depot$ git status
Sur la branche master

Aucun commit

Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure
dans ce qui sera validé)

    README.md

aucune modification ajoutée à la validation mais
des fichiers non suivis sont présents (utilisez
"git add" pour les suivre)
```

```
git status -s
```

```
?? README.md
```

Dépôt local

- Ajouter un fichier non suivi
 - **git add** [-i] <file paths | *>

```
matthieu@matthieu:~/depot$ git status
Sur la branche master

Aucun commit

Modifications qui seront validées :
  (utilisez "git rm --cached <fichier>..." pour
  désindexer)

    nouveau fichier : README.md
```

A README.md

<https://git-scm.com/docs/git-add>

Dépôt local

- Valider un contenu indexé dans le dépôt
 - **git commit -m <message>**

```
matthieu@matthieu:~/depot$ git commit -m "first commit"
[master (commit racine) 3f04de1] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

Sur la branche master
rien à valider, la copie de travail est propre

le fichier est à l'état **U : Unmodified**

Le fichier est dans le même état que dans le dépôt

Dépôt local

- Historique des commits
 - **git log** : Affichage des métadonnées des commits

```
matthieu@matthieu:~/depot$ git log
commit 3f04de1845e815c8ca094e2c8507b89dfe53ffb4 (HEAD -> master)
Author: mlamamra <mlamamra@jehann.fr>
Date:   Sun May 30 14:06:37 2021 +0200

    first commit
```

Dépôt local

- Historique des commits
 - **git log -p -<n>** : Affichage des diffs des n derniers commits

```
matthieu@matthieu:~/depot$ git log -p -1
commit 3f04de1845e815c8ca094e2c8507b89dfe53ffb4 (HEAD -> master)
Author: mlamamra <mlamamra@jehann.fr>
Date:   Sun May 30 14:06:37 2021 +0200

    first commit

diff --git a/README.md b/README.md
new file mode 100644
index 0000000..54ac958
--- /dev/null
+++ b/README.md
@@ -0,0 +1 @@
+# TITLE
```

Dépôt local

- Détailler un commit
 - **git show <commit>**

```
matthieu@matthieu:~/depot$ git show b2c716d
commit b2c716d59b110e134e99122fb8e2aced2e0cdb7a
Author: mlamamra <mlamamra@dawan.fr>
Date:   Sun May 30 16:27:08 2021 +0200
```

```
    git rm wrong_file
```

```
diff --git a/wrong_file b/wrong_file
deleted file mode 100644
index e69de29..0000000
```

Dépôt local

- Description d'un commit

```
git cat-file -p 98ca9
```

```
98ca9
commit size
tree 92ec2
author Scott
committer Scott
The initial commit of my project
```

```
git ls-tree -r 92ec2
```

```
92ec2
tree size
blob 5b1d3 README
blob 911e7 LICENSE
blob cba0a test.rb
```

5b1d3
blob size
== Testing library
This library is used to test Ruby projects.

911e7
blob size
The MIT License
Copyright (c) 2008 Scott Chacon
Permission is hereby granted, free of charge, to any person

cba0a
blob size
require 'logger'
require 'test/unit'
class Test::Unit::TestCase

blob : Binary Large Object – contenu d'un fichier
tree : représentation du contenu de l'index enregistré au moment du commit

blob, tree, et commit sont des **objets git**

Dépôt local

- Description d'un objet

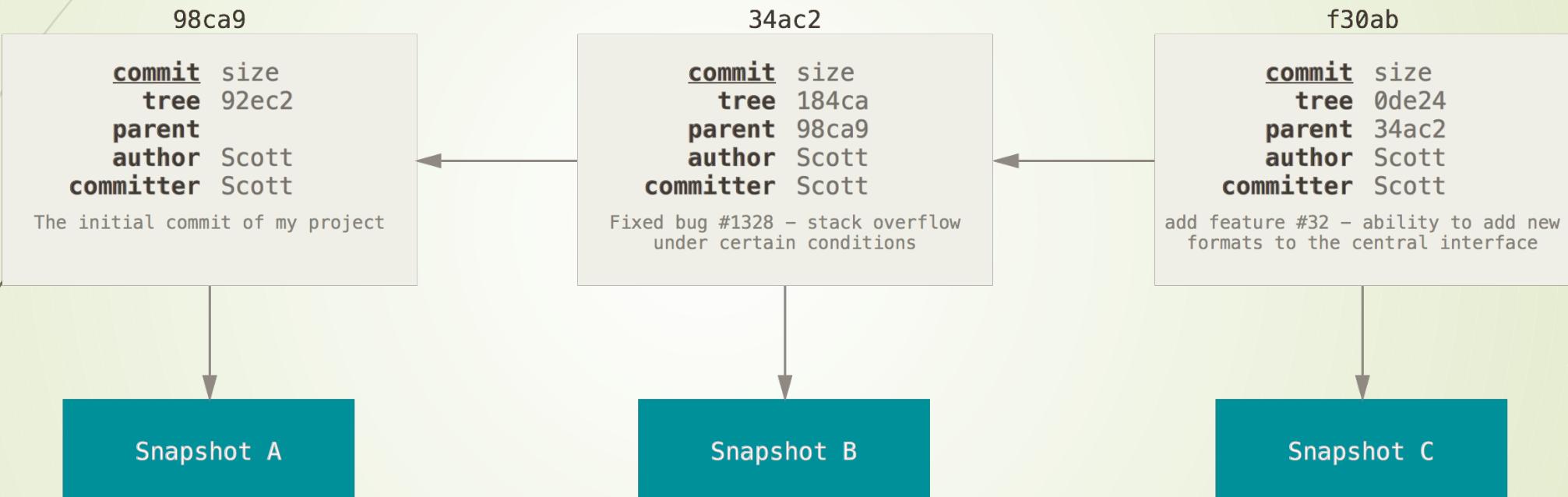
```
.git/objects/
├── 0d
│   └── f36b9c297ac667d49880895eecb1d3f9a8636d
├── 1c
│   └── badfb18e020151e12518ae6ec742682a0ae0d8
└── 72
    └── 7b9e88e168fe54c6e7876e980acdf1bec4a658
├── info
└── pack
```

blob content= zlib(header(blob filesize\0) + filecontent)
// name = sha1()
tree : sha1 à partir du contenu

commit : sha1 à partir du contenu

Dépôt local

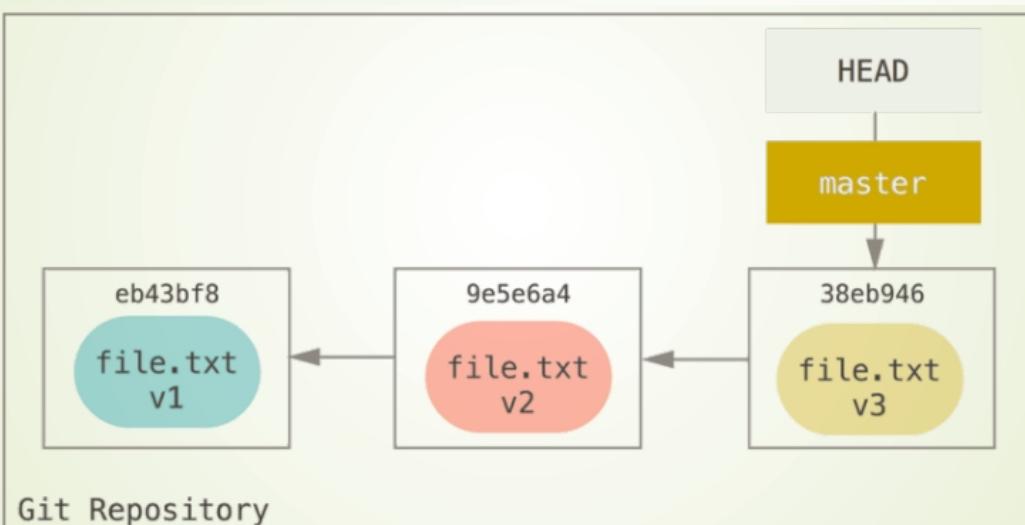
- enchaînement des commits



Dépôt local

- Pointeur de commit HEAD

HEAD est un pointeur désignant une **référence du dépôt**, qui pointe elle même sur un commit de la branche courante (par défaut, le commit le plus récent)



```
matthieu@matthieu:~/depot$ cat .git/HEAD
ref: refs/heads/master
```

```
matthieu@matthieu:~/depot$ cat .git/refs/heads/master
42773876719f02e012b5bfa05f2b3ebfa7d6ffa6
```

Dépôt local

- Modifier un fichier versionné

```
matthieu@matthieu:~/depot$ git status
Sur la branche master
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git checkout -- <fichier>..." pour annuler les modifications dans la copie de travail)

  modifié :      README.md
```

M README.md

Dépôt local

- Modifier un fichier versionné
 - **git diff <file>** : voir les modifications

```
matthieu@matthieu:~/depot$ git diff README.md
diff --git a/README.md b/README.md
index 54ac958..6865f66 100644
--- a/README.md
+++ b/README.md
@@ -1 +1,3 @@
 # TITLE
+
+## Sub Title
\ No newline at end of file
```

<https://git-scm.com/docs/git-diff>

Dépôt local

- Valider un contenu à l'état Modifié
 - **git commit -am** <message> : a pour add

Sur la branche master
rien à valider, la copie de travail est propre

Dépôt local

- Le fichier `.gitignore`
 - Rendre certains fichiers / dossiers **invisibles pour git**

```
.git/  
.gitignore  
LICENSE  
README.md  
venv/
```



```
READ*  
venv/  
!README.md
```



```
A LICENSE  
?? .gitignore
```

* : tous les caractères
/ : dossier et son contenu
** : à travers les sous dossiers
! : exception à une règle plus haut

Dépôt local

- Etat de la copie de travail

À un instant donné, l'état de la copie de travail est composé :

- Des fichiers **Unmodified** dans le même état que **HEAD**
- Plus les fichiers **Modified** ou **Ajoutés** à l'index
- Plus les fichiers **Untracked / non suivis** (**??**)

```
.git/  
.gitignore  
LICENSE  
README.md  
venv/
```

```
A LICENSE  
?? .gitignore
```

```
matthieu@matthieu:~/depot$ git log --pretty=short -1  
commit c27bce97bf908c0d3c2a19ab5ceabd3820c73f1c (HEAD -> master)  
Author: mlamamra <mlamamra@dawan.fr>
```

subtitle

Dépôt local

- remettre un fichier modifié dans l'état du dépôt
 - **git checkout [<tree-ish>] -- <file>**

```
A LICENSE
M README.md
?? .gitignore
```

- **tree-ish** : HEAD (défaut), commit, branche, refs ...

```
matthieu@matthieu:~/depot$ git checkout HEAD -- README.md
matthieu@matthieu:~/depot$ git status -s
A LICENSE
?? .gitignore
```

- Les modifications sont perdues
- **--** est utilisé pour séparer certains paramètres dans les commandes

Dépôt local

- désindexer un fichier par suppression
 - **git rm --cached [-r]<file> | <dir>**

```
matthieu@matthieu:~/depot$ git rm --cached LICENSE
rm 'LICENSE'
matthieu@matthieu:~/depot$ git status -s
?? .gitignore
?? LICENSE
```

- Si le fichier était déjà dans le dépôt un commit l'effacera du dépôt

Dépôt local

- désindexer un fichier par mise à jour de l'index
 - **git reset [--mixed] [<tree-ish>] -- <file>**

```
matthieu@matthieu:~/depot$ git reset HEAD -- LICENSE
matthieu@matthieu:~/depot$ git status -s
?? .gitignore
?? LICENSE
```

- Si le fichier était déjà dans le dépôt ce reset n'a pas d'effet
- Les modifications en cours sont sauvegardées (cf git reset)

<https://git-scm.com/docs/git-reset>

Dépôt local

- Supprimer un fichier du disque et du dépôt
 - **git rm [-r] <file> | <dir>**

```
.git/  
.gitignore  
LICENSE  
README.md  
wrong_file
```

add + commit



```
git log --pretty="%h %s" -1
```

```
4efa31f commit wrong_file
```

```
.git/  
.gitignore  
LICENSE  
README.md
```

status



```
D wrong_file
```

commit



```
b2c716d git rm wrong_file
```

Dépôt local

- Renommer un fichier sur le disque et dans le dépôt
 - **git mv <oldname> <newname>**

```
.git/  
.gitignore  
LICENSE  
README.md  
wrong_name
```

add + commit

```
git log --pretty="%h %s" -1  
a2add8a commit wrong_name file
```

```
.git/  
.gitignore  
good_name  
LICENSE  
README.md
```

status

```
R wrong_name -> good_name
```

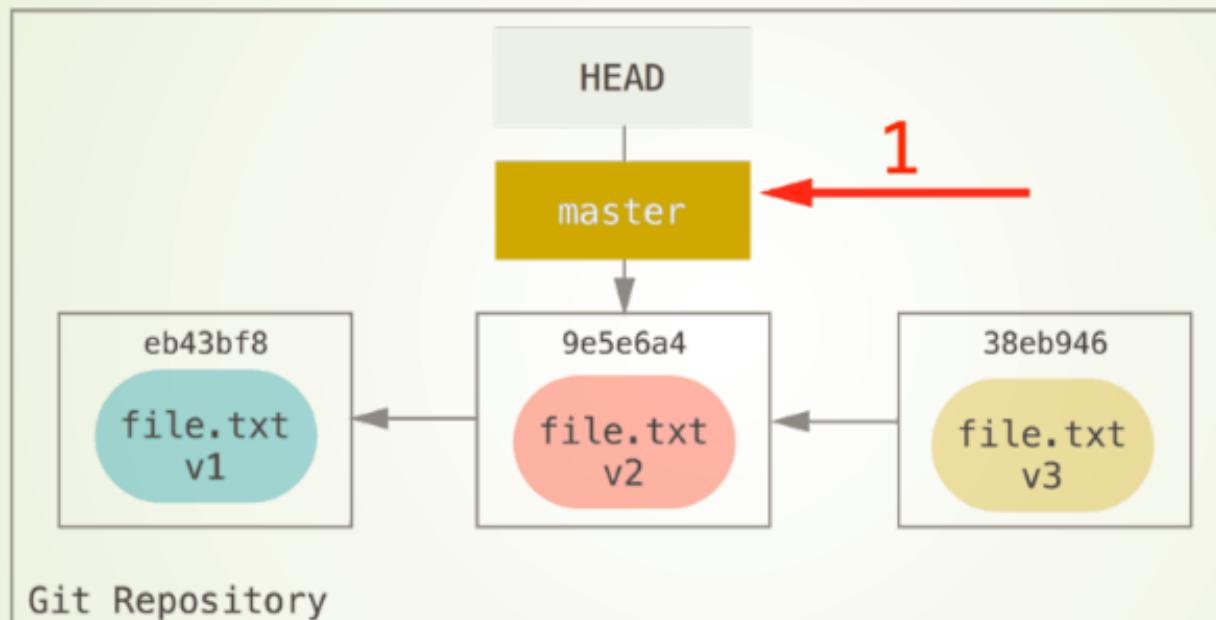
Git mv ~ = mv oldname newname + git rm oldname + git add newname

commit

```
6ccfc2f git mv
```

Dépôt local

- Déplacer le Pointeur HEAD : **git reset**



git reset HEAD~1

« Rembobinage : »

git reset HEAD@{1}

git reset HEAD@{n} : déplacement dans l'énième position de HEAD précédent la position actuelle

git reflog [show HEAD] : historique des déplacements du HEAD

Dépôt local

- Types de reset

Option	Dépôt	Index	Copie de travail
<code>git reset --soft < tree-ish ></code>	écrasé	conservé	conservée
<code>git reset [--mixed] < tree-ish ></code>	écrasé	écrasé	conservée
<code>git reset --hard < tree-ish ></code>	écrasé	écrasé	écrasée

Dépôt local

```
git log --oneline -1
```

```
4277387 (HEAD -> master) README
```

```
b2c716d (HEAD -> master) git rm wrong_file
```

```
A LICENSE  
M README.md  
?? .gitignore
```

git reset --soft HEAD~1



```
A LICENSE  
MM README.md  
?? .gitignore
```

git reset HEAD~1



```
M README.md  
?? .gitignore  
?? LICENSE
```

git reset --hard HEAD~1



```
?? .gitignore  
?? LICENSE
```

```
matthieu@matthieu:~/depot$ git reset --hard HEAD~1  
HEAD est maintenant à b2c716d git rm wrong_file
```

Dépôt local

- Annuler un commit précédent
 - **git revert [--no-edit (-e) | --no-commit] <tree-ish>**
 - Création d'un **nouveau commit** annulant les modifications d'un commit

```
matthieu@matthieu:~/depot$ printf "\n * new content\n" >> README.md && \
> git commit -am "new content"
[master f17881d] new content
 1 file changed, 2 insertions(+)
matthieu@matthieu:~/depot$ git log --oneline -2
f17881d (HEAD -> master) new content
0819cf1 fixing README
matthieu@matthieu:~/depot$ git revert HEAD
[master 0f8e7c8] Revert "new content"
 1 file changed, 2 deletions(-)
matthieu@matthieu:~/depot$ git log --oneline -2
0f8e7c8 (HEAD -> master) Revert "new content"
f17881d new content
matthieu@matthieu:~/depot$ cat README.md
# TITLE

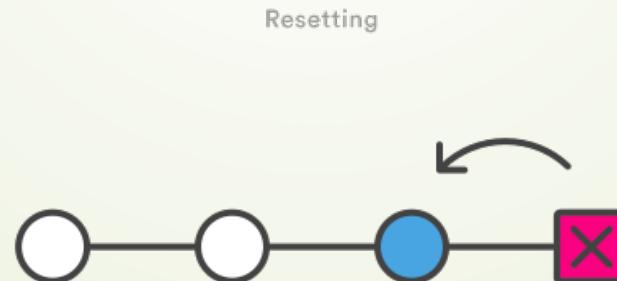
## Sub Title Fixed
```

Dépôt local

- reset vs revert
 - **git revert ne modifie pas l'historique des logs**

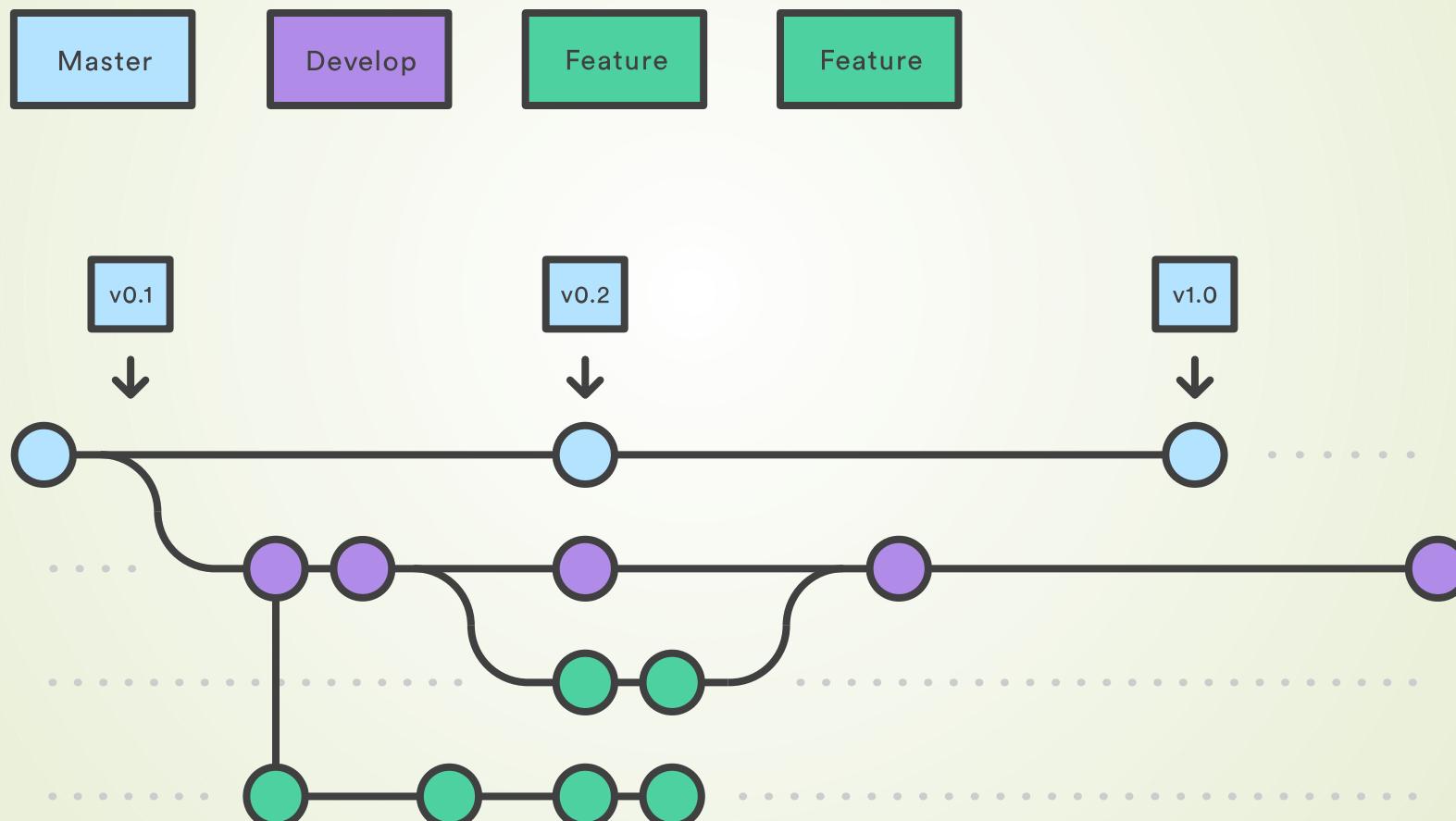


- **Un commit après un reset peut modifier l'historique**



Les branches

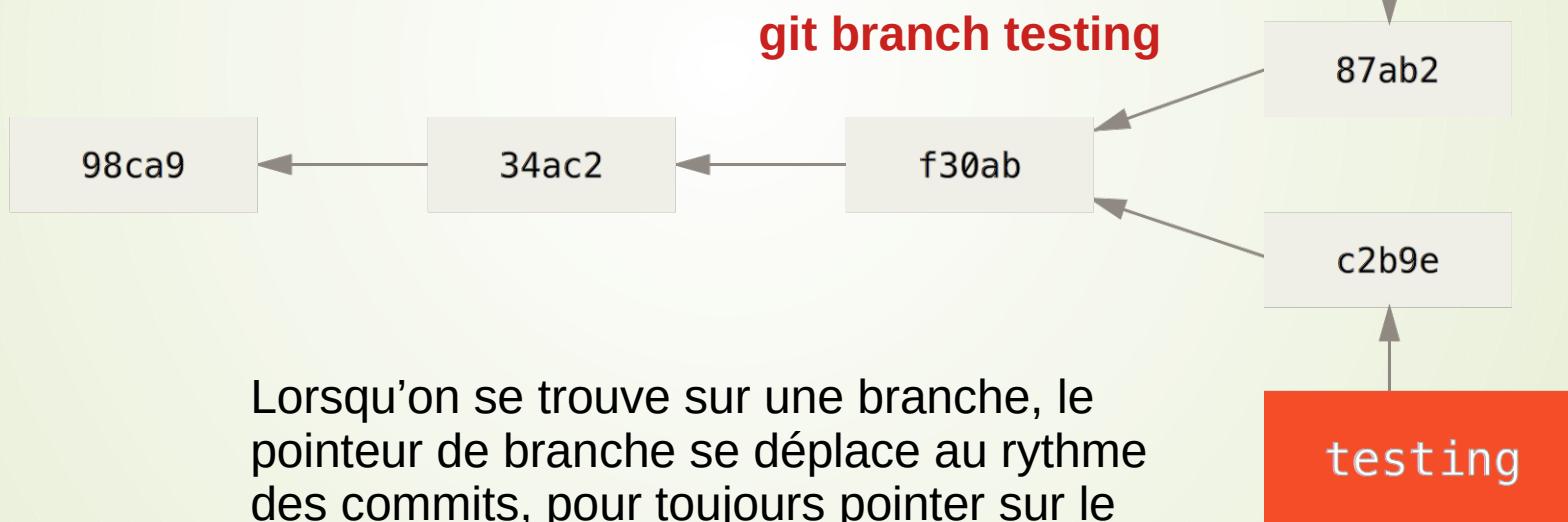
- Parallélisation des développements



Les branches

- Description des branches

- Une branche est un pointeur créé sur un commit particulier
- La branche initiale se nomme « master » ou « main »
- le pointeur **HEAD** pointe toujours sur un commit de la branche courante



Les branches

- Crédation / Basculement de branche
 - **git branch <branchname>**

```
git branch feature1
```

- **git checkout <branchname>**

```
matthieu@matthieu:~/depot$ git checkout feature1
Basculement sur la branche 'feature1'
```

- **git branch (-l | -v)**

```
matthieu@matthieu:~/depot$ git branch -v
* feature1 0f8e7c8 Revert "new content"
  master    0f8e7c8 Revert "new content"
```

- **git checkout -b <branchname>** : Crédation et basculement

Les branches

- Historique des commits :
 - **git log --graph**

```
matthieu@matthieu:~/depot$ git log --oneline --graph
* 0f8e7c8 (HEAD -> feature1, master) Revert "new content"
* f17881d new content
* 0819cf1 fixing README
* b2c716d git rm wrong_file
* 4efa31f commit wrong_file
* c3905ed desindexer LICENSE
* d983fa2 git rm
* 6ccfc2f git mv
```

Les branches

- Voir les divergences : **git log --graph --all**

```
matthieu@matthieu:~/depot$ git log --oneline --graph --all
* 803d99f (master) 2ème modif master
* 2fbdfb2 first modif master
| * fa799fc (HEAD -> feature1) 2nd modif feature 1
| * 64c2249 first modif feature 1
|/
* 0f8e7c8 Revert "new content"
* f17881d new content
```

Les branches

- Fusion de branches
 - **git merge [--no-edit (-e)] <branchname>**

```
matthieu@matthieu:~/depot$ git checkout master
Basculement sur la branche 'master'
matthieu@matthieu:~/depot$ git merge feature1
Merge made by the 'recursive' strategy.
 README.md | 2 ++
 1 file changed, 2 insertions(+)
```

- On se place sur la branche qui reçoit la fusion
- on rapatrie les modifications contenues dans les commits de l'autre branche
- git ouvre un éditeur pour message

```
* 5ba767f (HEAD -> master) Merge branch 'feature1'
|\ 
| * fa799fc (feature1) 2nd modif feature 1
| * 64c2249 first modif feature 1
* | 803d99f 2ème modif master
* | 2fbdfb2 first modif master
|| 
* 0f8e7c8 Revert "new content"
```

- Création d'un **commit de fusion**

Les branches

- Suppression de branches

Possible tant qu'il n'y a pas de nouveaux commits dessus ou quand elle est fusionnée

```
matthieu@matthieu:~/depot$ git branch -d feature1  
Branche feature1 supprimée (précédemment fa799fc).
```

Posibilité de forcer la suppression de branche

git branch -D <branch_name> => perte des commits non fusionnés de la branche

Les tags

- Lightweight tags

Un tag léger est un **pointeur** sur un commit particulier

Ex : **git tag v1.0 c02e9a4** { « v1.0 » : nom du tag, « c02e9a4 » : id du commit (default HEAD) }

Liste des Tags / description d'un tag :

```
matthieu@matthieu:~/Documents/git_formation/depot$ git show v1.0
commit c02e9a4ba9f9e04bcfedffffd3724f3c22a5424d2 (HEAD -> alternatef5, tag:
  v1.0)
Author: mlamamra <mlamamra@jehann.fr>
Date:   Wed Oct 14 12:15:14 2020 +0200

  content19

diff --git a/fic1 b/fic1
index 5d3b8d4..3c09727 100644
--- a/fic1
+++ b/fic1
@@ -6,4 +6,4 @@ content15
  content16
  content17
  content18
-
+content19
```

Les tags

- Annotated tags

Un tag annoté rajoute des **métadonnées** dont un **message** du créateur du tag

Ex : **git tag -a v1.1 -m « page d'accueil »**

description d'un tag annoté :

```
matthieu@matthieu:~/Documents/git_formation/depot$ git show v1.1
tag v1.1
Tagger: mlamamra <mlamamra@jehann.fr>
Date:   Wed Oct 14 14:21:44 2020 +0200

page d'accueil

commit f06f6e4cfb78f4873f7ea182b57da5573af35abf (HEAD -> alternatef5, tag:
v1.1)
Author: mlamamra <mlamamra@jehann.fr>
Date:   Wed Oct 14 14:21:04 2020 +0200

    content19'
```

Les tags

- suppression

En local

ou -d

```
matthieu@matthieu:~/Documents/git_formation/depot$ git tag --delete v1.1
Étiquette 'v1.1' supprimée (elle était sur fc2787f)
```

Suppression d'un tag distant

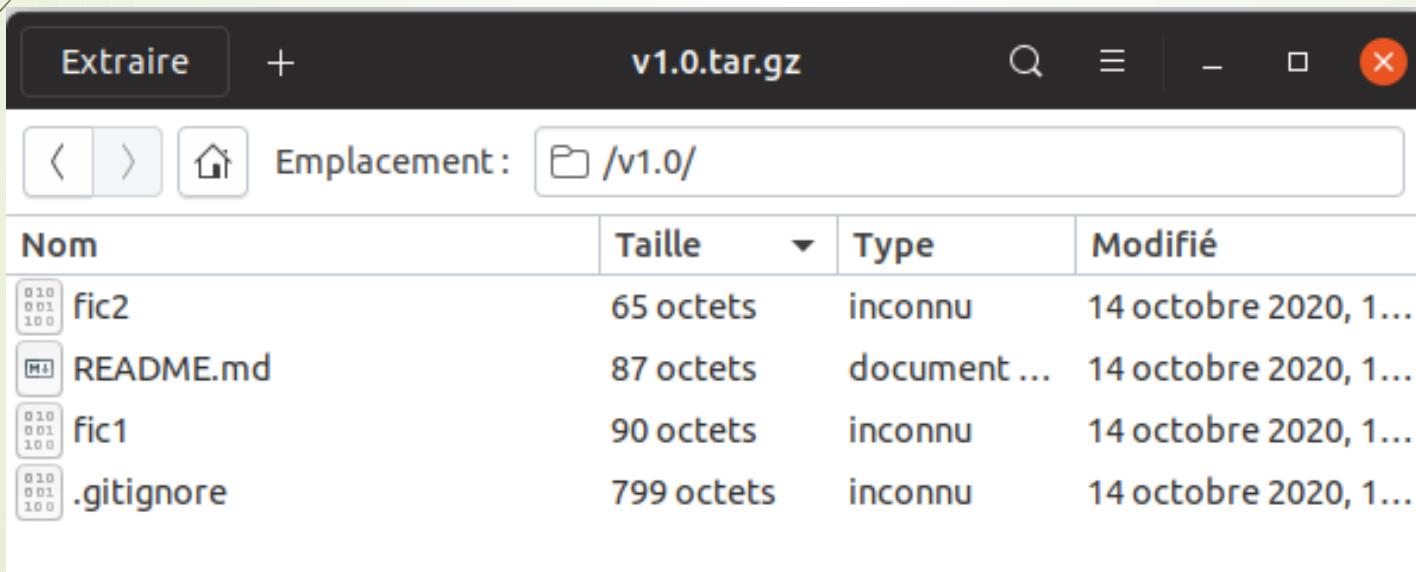
```
matthieu@matthieu:~/Documents/git_formation/depot$ git push origin --delete v1.1
Username for 'https://github.com': lamamram
Password for 'https://lamamram@github.com':
To https://github.com/lamamram/formation.git
 - [deleted]           v1.1
```

Les tags

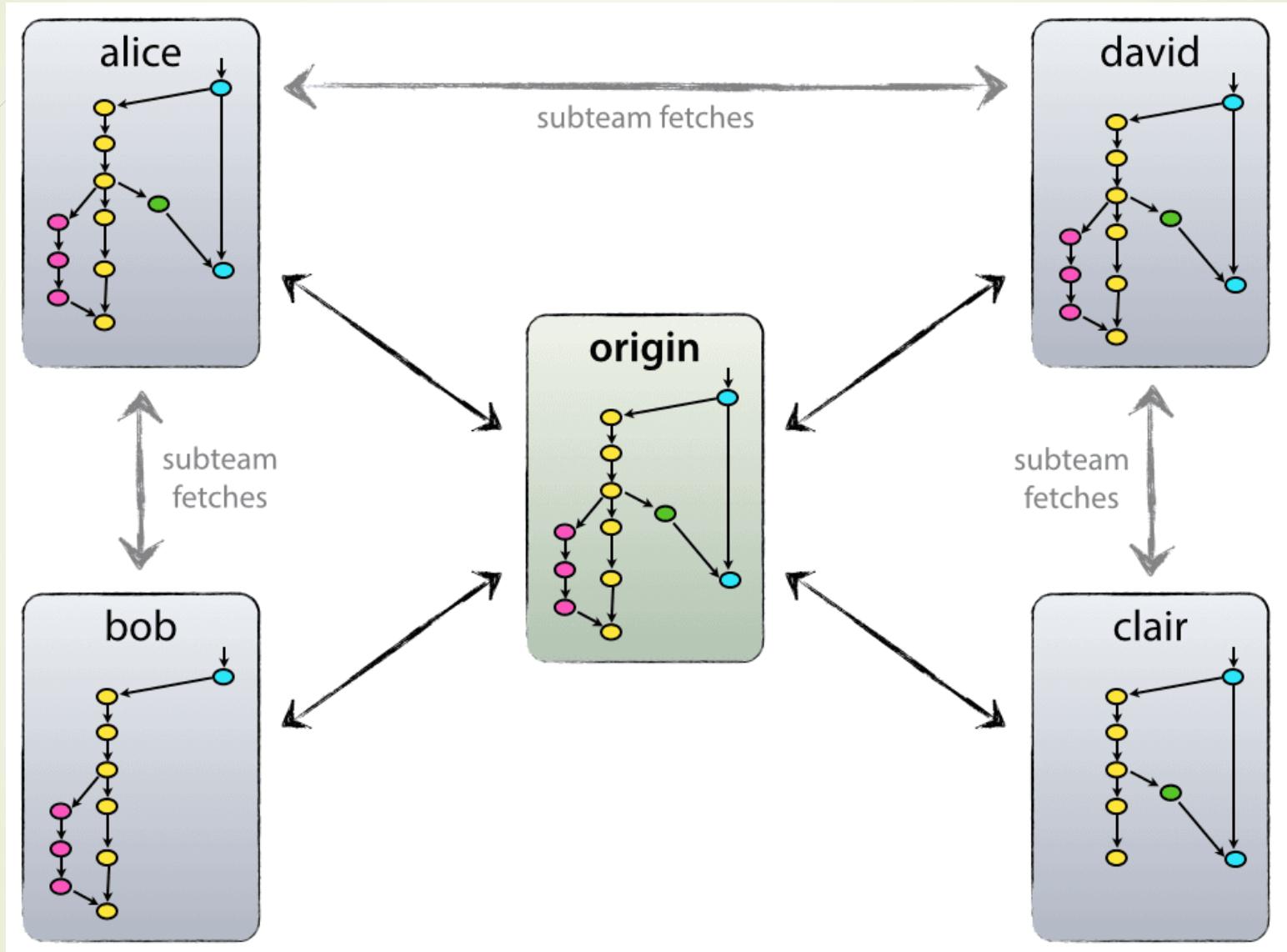
- export (pour déploiement)

```
git archive --prefix=v1.0/ -o v1.0.tar.gz v1.0
```

-prefix permet de packager les fichiers dans un dossier
-o se charge d'opérer la compression demandée en extension



Dépôts distants



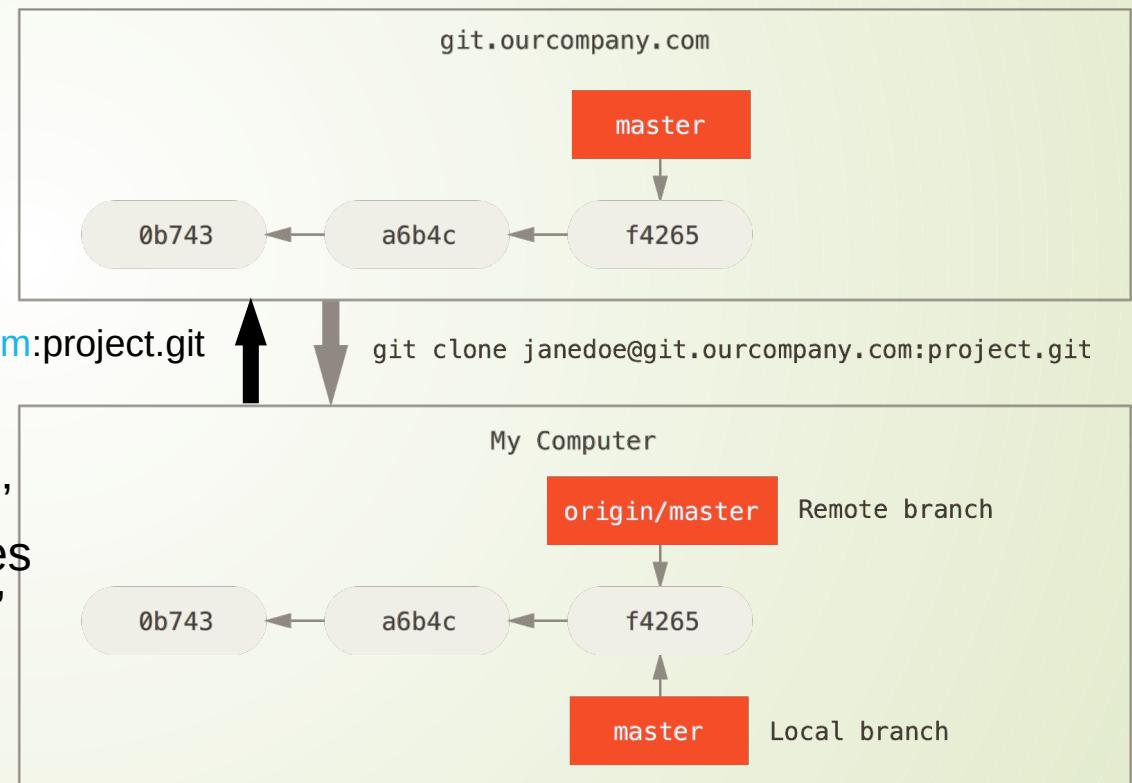
Dépôt distant

- Connexions à un dépôt distant

git clone : copie d'un dépôt distant avec ses branches

git remote add origin janedoe@git.ourcompany.com:project.git
git pull origin master

git remote : connexion à un dépôt distant “**origin**”
git pull origin master : rapatriement en local du des derniers commits de la branche master de “origin”



Dépôt distant

- Ajouter un dépôt distant au dépôt local
- **git remote add** <local_name> <https://[url dépôt] || [user]@[domain || ip]:/[path/to/repo.git]>

```
git remote add origin git@gitlab.myusine.fr:root/myusine.git
```

```
matthieu@matthieu:~/depot$ git remote -v
origin  git@gitlab.myusine.fr:root/myusine.git (fetch)
origin  git@gitlab.myusine.fr:root/myusine.git (push)
```

git fetch : opération de rapatriement d'une branche distante en locale (en lecture seule)

git push : opération d'envoi de données sur une branche distante depuis une branche locale

Dépôt distant

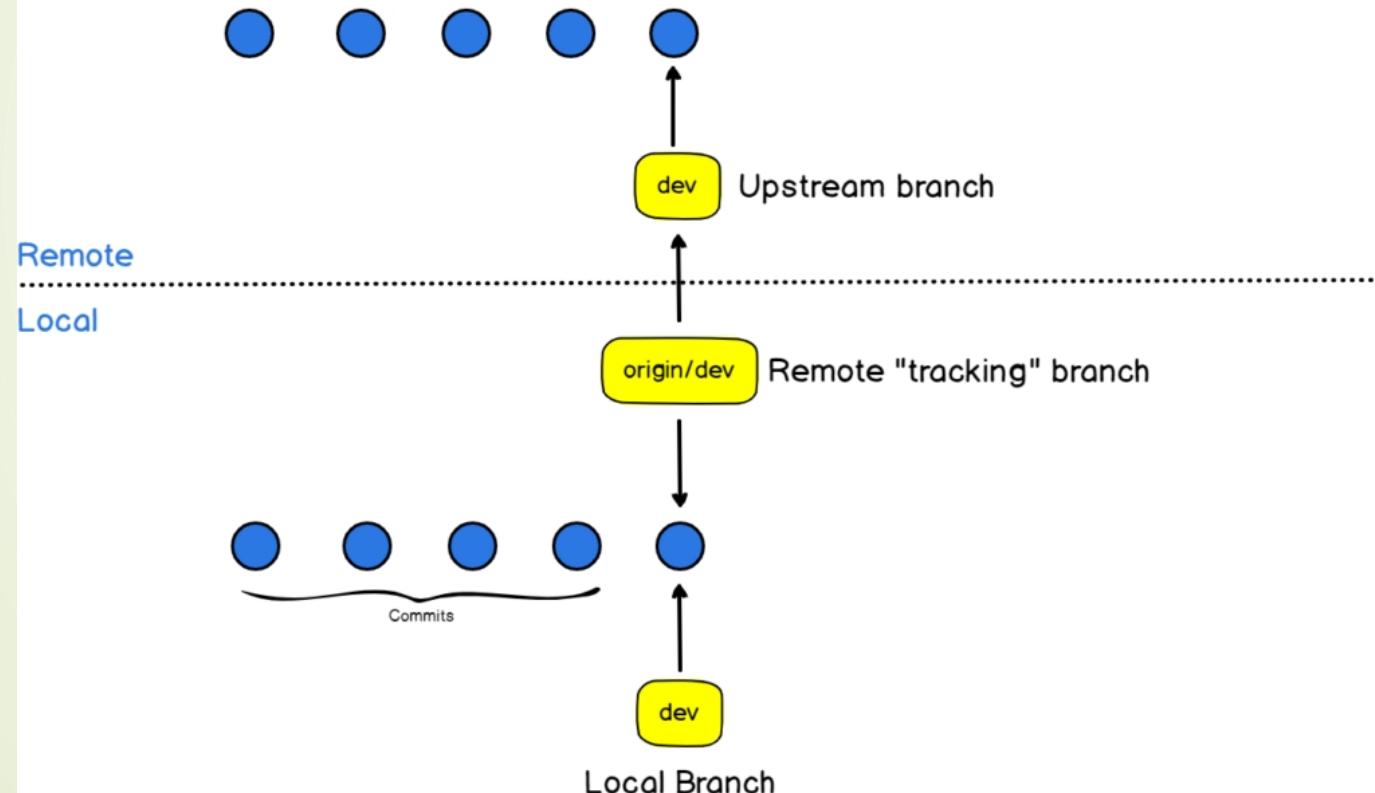
- Transférer des commits sur un dépôt distant
 - `git push < repository > < branchname >`

```
matthieu@matthieu:~/depot$ git push origin master
Énumération des objets: 42, fait.
Décompte des objets: 100% (42/42), fait.
Compression par delta en utilisant jusqu'à 4 fils d'exécution
Compression des objets: 100% (28/28), fait.
Écriture des objets: 100% (42/42), 3.66 KiB | 1.83 MiB/s, fait.
Total 42 (delta 5), réutilisés 0 (delta 0)
To gitlab.myusine.fr:root/myusine.git
 * [new branch]      master -> master
```

Dépôt distant

- Suivi de branches entre dépôt

Upstream branches explained



Dépôt distant

- Suivi de branches entre dépôt
 - `git push (-u | --set-upstream) <repository> <branchname>`

```
git push -u origin HEAD
```

- **HEAD** => la branche distante porte le même nom qu'en local

La branche 'master' est paramétrée pour suivre la branche distante 'master' depuis 'origin'.

```
matthieu@matthieu:~/depot$ git branch -vv
* master 2319c41 [origin/master] test branches de suivi
```

- Avec l'upstream on peut déclencher `git push` (et `git pull`) sans paramètres
- L'upstream permet également de connaître la position du dépôt local par rapport au dépôt distant en terme de commits

Dépôt distant

- Cloner un dépôt distant

git clone : réplication exacte d'un dépôt distant en local

```
matthieu@matthieu:~/depot2$ git clone git@gitlab.myusine.fr:root/myusine.git .
Clonage dans '.'...
remote: Enumerating objects: 45, done.
remote: Counting objects: 100% (45/45), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 45 (delta 7), reused 0 (delta 0), pack-reused 0
Réception d'objets: 100% (45/45), fait.
Résolution des deltas: 100% (7/7), fait.
```

Dépôt distant

- Git pull

```
matthieu@matthieu:~/Documents/git_formation/depot2$ git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 2), reused 3 (delta 2), pack-reused 0
Dépaquetage des objets: 100% (3/3), fait.
Depuis https://github.com/lamamram/formation
 * branch            master      -> FETCH_HEAD
   7010ff3..7c9bf27  master      -> origin/master
Mise à jour 7010ff3..7c9bf27
Fast-forward
  fic1 | 2 ++
  1 file changed, 2 insertions(+)
```

Pull = fetch (origin/master branche distante) + merge (master → origin/master)

Dépôt distant

- push d'un tag sur un dépôt distant

```
matthieu@matthieu:~/Documents/git_formation/depot$ git push origin v1.1
Username for 'https://github.com': lamamram
Password for 'https://lamamram@github.com':
Énumération des objets: 68, fait.
Décompte des objets: 100% (68/68), fait.
Compression par delta en utilisant jusqu'à 4 fils d'exécution
Compression des objets: 100% (55/55), fait.
Écriture des objets: 100% (66/66), 5.54 KiB | 405.00 KiB/s, fait.
Total 66 (delta 21), réutilisés 0 (delta 0)
remote: Resolving deltas: 100% (21/21), done.
To https://github.com/lamamram/formation.git
 * [new tag]           v1.1 -> v1.1
```

Une fois créés, Les tags sont indépendants de la branche dont ils sont issus

Dépôt distant

- Crédit d'une branche distante

```
matthieu@matthieu:~/Documents/git_formation/depot2$ git checkout -b feature7
Basculement sur la nouvelle branche 'feature7'
matthieu@matthieu:~/Documents/git_formation/depot2$ nano fic2
matthieu@matthieu:~/Documents/git_formation/depot2$ git commit -am "content29"
[feature7 3f31603] content29
 1 file changed, 1 insertion(+), 1 deletion(-)
matthieu@matthieu:~/Documents/git_formation/depot2$ git push origin feature7
Username for 'https://github.com': lamamram
Password for 'https://lamamram@github.com':
Énumération des objets: 5, fait.
Décompte des objets: 100% (5/5), fait.
Compression par delta en utilisant jusqu'à 4 fils d'exécution
Compression des objets: 100% (3/3), fait.
Écriture des objets: 100% (3/3), 277 bytes | 277.00 KiB/s, fait.
Total 3 (delta 2), réutilisés 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'feature7' on GitHub by visiting:
remote:     https://github.com/lamamram/formation/pull/new/feature7
remote:
To https://github.com/lamamram/formation.git
 * [new branch]      feature7 -> feature7
```

Dépôt distant

- récupération d'une branche distante
- **git checkout --track <upstream_branch>** permet de créer une branche locale à partir d'une branche de suivi récupérée depuis le dépôt distant avec **git fetch**

```
matthieu@matthieu:~/Documents/git_formation/depot$ git fetch origin feature7
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 2), reused 3 (delta 2), pack-reused 0
Dépaquetage des objets: 100% (3/3), fait.
Depuis https://github.com/lamamram/formation
 * branch            feature7    -> FETCH_HEAD
 * [nouvelle branche] feature7    -> origin/feature7
matthieu@matthieu:~/Documents/git_formation/depot$ git checkout --track origin/feature7
La branche 'feature7' est paramétrée pour suivre la branche distante 'feature7' depuis 'origin'.
Basculement sur la nouvelle branche 'feature7'
```

Dépôt distant

- suppression

Suppression d'une branche distante

Suppression d'un tag distant

```
matthieu@matthieu:~/Documents/git_formation/depot$ git push origin --delete v1.1
Username for 'https://github.com': lamamram
Password for 'https://lamamram@github.com':
To https://github.com/lamamram/formation.git
 - [deleted]           v1.1
```

REBASING

- Modifier l'historique des branches

```
* dab564e README10  (HEAD -> master)
| *
| * 3514211 content113' (feature7)
| * b918dcc content30
| * 3f31603 content29 (origin/feature7)
|| /
```

```
git checkout feature7
git rebase master
```

```
* 8205849 content113' (HEAD -> feature7)
* d6fb331 content30
* 9be8d97 content29
* dab564e README10 (master)
| * 3f31603 content29 (origin/feature7)
|| /
```