Utilisation locale

Romain THERRAT

POCKOST

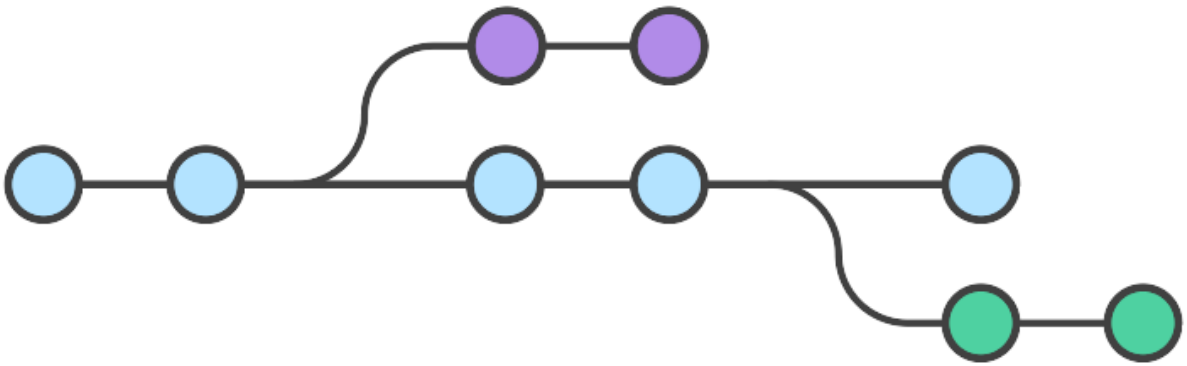
14 décembre 2020

Sommaire

Qu'est-ce qu'une branche ?

- Divergence dans le code
- Nouvelle fonctionnalité = copie du code
- C'est le cas de `SVN`
- `git` apporte une autre approche
  - Simple
  - Évite la duplication
  - Beaucoup plus léger (espace disque)
- À utiliser le plus possible !

Les branches



Comment fonctionnent les branches ?

Une branche c'est

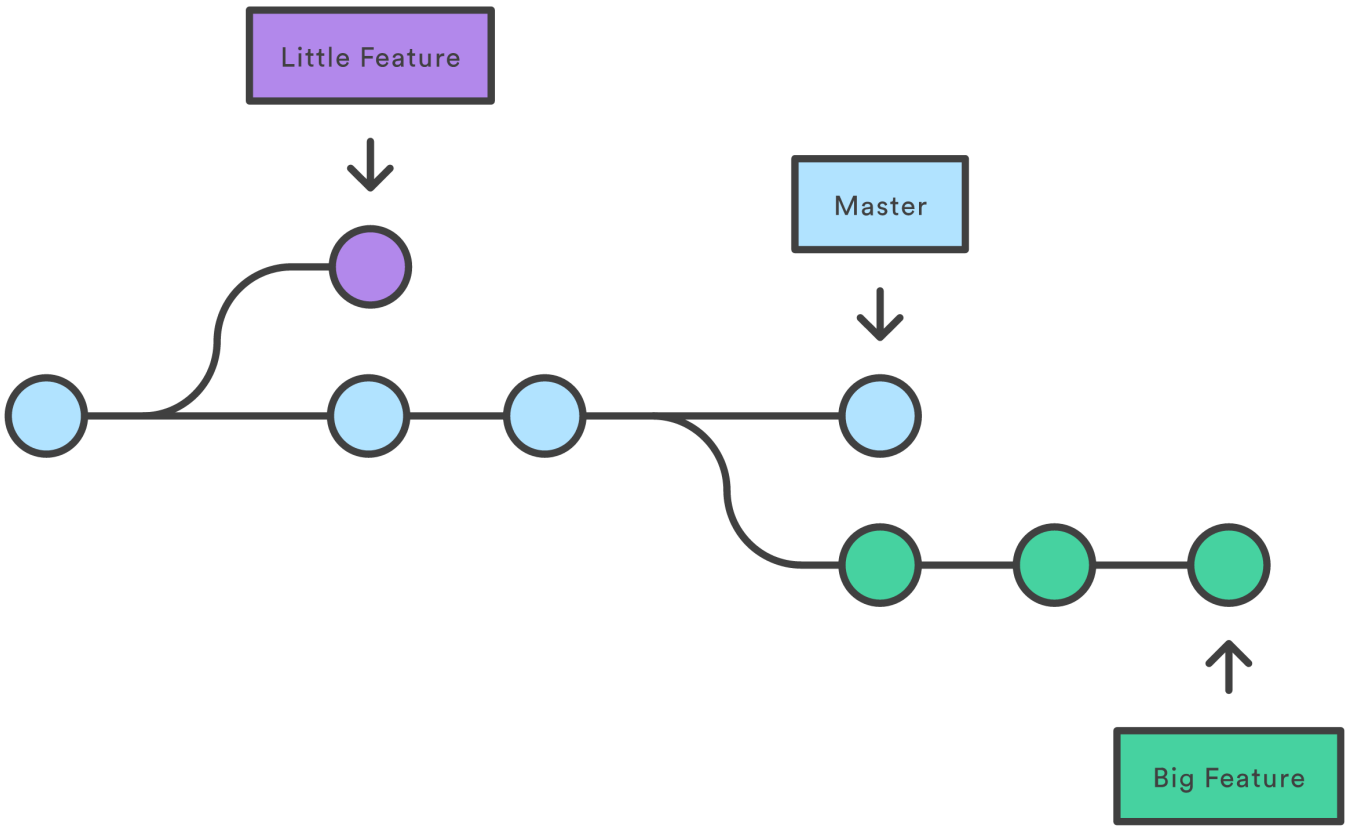
- Un nom
- Un numéro de commit

Et par défaut

- Nom `master`
- Une nouvelle branche = commit courant

Donc passer d'une branche à l'autre ne revient qu'à afficher un commit particulier

Les branches



Utilisation des branches

- Voir la branche courante
- `git status`
- Créer une nouvelle branche
- `git branch nom` ou `git checkout -b nom`
- Changer de branche
- `git checkout nom`
- Note : Il peut être impossible de changer de branche si vous avez des modifications non commitées qui entre en conflit avec la branche de destination.
- Nous verrons en détail les fusions et cie dans le prochain épisode.

git branch

```
$ git branch dev
$ git checkout dev
Switched to branch 'dev'
$ git status
On branch dev
nothing to commit, working tree clean
$ git checkout master
Switched to branch 'master'
```

Qu'est-ce qu'un tag

- Version d'un site
  - v1.0
  - v2.0
  - v2.1
  - ...
- Un nom sur un commit
- Commande `git tag`

git tag

Voir les tags

```
$ git tag
v1.0
v1.1
```

Créer un tag pour le commit courant

```
$ git tag v2.1
```

Se placer sur un tag

```
$ git checkout v1.1
```

Sommaire

Historique

Nous avons vu dans le chapitre précédent qu'il était possible

- De consulter l'historique des changements
- Se déplacer d'une branche à l'autre

Ceci est possible grâce au fonctionnement des commits. Aussi il est aussi possible de

- Se placer dans un état particulier (=sur un commit)
- Inverser un commit
- Et même . . . supprimer un commit !

Se placer sur un commit

Pour se placer sur un commit nous utilisons la commande
`git checkout <id>`

```
git checkout d515e5c2b0b2b2ba5797cbfd996f0da2a4f72c6d
Note: checking out 'd515e5c2b0b2b2ba5797cbfd996f0da2a4f72c6d'

You are in 'detached HEAD' state. You can look around, make
changes and commit them, and you can discard any commits you
state without impacting any branches by performing another c

If you want to create a new branch to retain commits you cre
do so (now or later) by using -b with the checkout command a

  git checkout -b <new-branch-name>

HEAD is now at d515e5c Second commit
```

Revert

Pour inverser un commit nous utilisons `git revert <id>`. Ceci crée un nouveau commit

```
git revert bebc26f99eeca85f8b4d24ac6802b82d4ec24dca
[master 0ea76d3] Revert "3de commit"
 3 files changed, 2 deletions(-)
 delete mode 100644 fichier3.txt
```

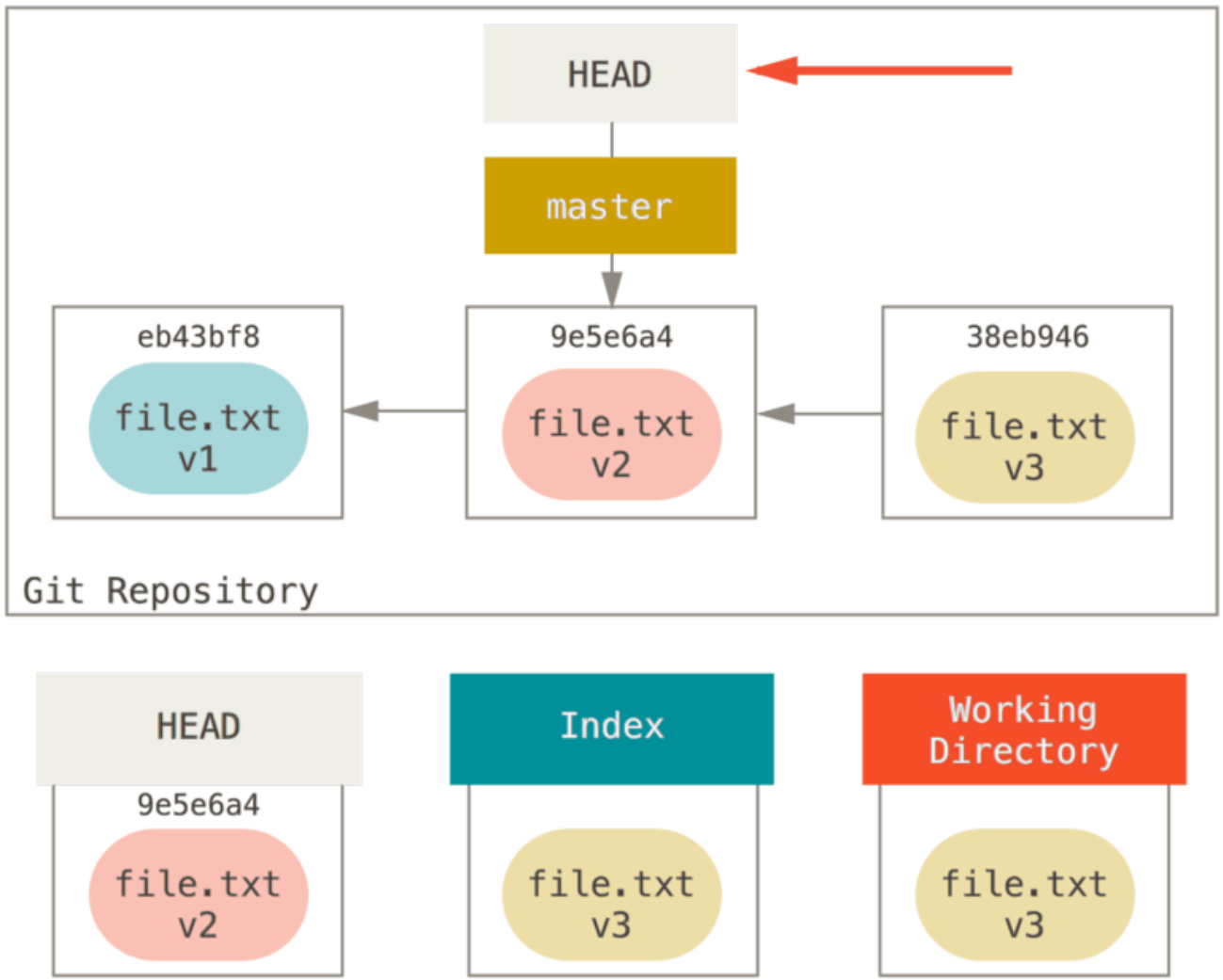
reset

Pour supprimer un ou plusieurs commits nous pouvons utiliser la commande `git reset`. On déplace `HEAD`

```
$ git reset d515e5c
HEAD is now at d515e5c Second commit
```

`git reset` est aussi utilisé pour sortir un fichier du `staging`

reset


HEAD
master
eb43bf8
file.txt
v1
9e5e6a4
file.txt
v2
38eb946
file.txt
v3
Git Repository
HEAD
9e5e6a4
file.txt
v2
Index
file.txt
v3
Working
Directory
file.txt
v3
git reset --soft HEAD~

reest soft et hard

- Un reset soft
  - Ne déplace que HEAD
  - Les modifications sont dans l'index
  - `git reset -soft <id>`
  - Option par défaut
- Un reset hard
  - Déplace HEAD
  - Remet les fichiers de l'index dans l'état de HEAD
  - `git reset -hard <id>`

Sommaire

Labs

Nous allons continuer notre projet précédent en ajoutant une notion de branche à celui-ci.

- Placer vous sur la branche master
- Créer une branche `develop` basé sur la dernière version de la branche master
- Placer vous dans cette branche et confirmer que vous vous situez bien dans celle-ci via `git status`
- Ajouter un fichier `contact.php` à votre projet et commiter le
- Constater la présence de ce fichier dans l'historique
- Déplacer vous sur la branche `master`
- Constater que votre commit n'est plus présent
- Afficher l'arbre git avec `gitk`

Labs

Nous allons maintenant utiliser des tags

- Tagguer le commit de la branche `master` en tant que `v1.0`
- Modifier le fichier `index.php` pour ajouter à la fin une ligne contenant ?>
- Commiter cette modification
- Tagguer ce nouveau commit sous le nom `v1.1`
- Afficher la liste des tags
- Placer vous sur le commit taggué en v1.0
- Consulter l'historique en CLI et en GUI

Labs

Nous allons maintenant supprimer un commit

- Placer vous sur `master`
- Visualiser l'historique
- Faire un `revert` du commit précédent (ajout d'une ligne dans index.php)
- Constater qu'un nouveau commit a été créé dans l'historique
- Afficher l'état de l'arbre dans `gitk`
- Utiliser la commande `git reset` pour annuler les deux précédents commit
- Afficher l'état de l'arbre dans `gitk`
- Constater que les tags existent toujours

Labs

Tous comme dans l'exercice précédent vous pouvez reproduire ces actions sur le site ▸ git-school.github.io/visualizing-git/.