Інструменти GIT

HTTPS://GIT-SCM.COM/

ПІДРУЧНИК

HANDBOOK

Вибір ревізій

Щоб переглянути окремий коміт, необхідно дізнатися хеш-код цього коміту (наприклад командою git log), а далі використати команду

```
$ git show <hash>
```

наприклад,

```
$ git show 8bd0d40710465dac088b18b634dd9ebfa191c532
```

покаже всю необхідну інформацію про коміт (повідомлення, автора, зміни, зроблені в коміті, тощо), хеш якого 8bd0d40710465dac088b18b634dd9ebfa191c532.

Можна вказувати не повний хеш, а лише його перші кілька символів, якщо вони однозначно ідентифікують коміт:

```
$ git show 8bd0d40
```

Детальніше...

Вибір ревізій (гілкові посилання)

Якщо не вказати хеш коміту, для якого треба показати ревізію, буде показано ревізію для останнього коміту поточної гілки

```
$ git show
```

Якщо після команди show вказати ім’я гілки, буде показано ревізію коміту на який вказує ця гілка

```
$ git show <branch>
```

наприклад,

```
$ git show develop
```

покаже ревізію останнього коміту (повідомлення, автора, зміни, зроблені в коміті, тощо), гілки develop

Детальніше...

Вибір ревізій (HEAD)

Звернемо увагу, що спеціальне ім'я HEAD вказує на останній коміт поточної гілки. Таким чином команда

```
$ git show HEAD
```

виведе ревізію останнього коміту, подібно до команди `git show` без параметрів і відповідно ніякого сенсу особливо немає. Проте, потрібно пам'ятати, що в GIT є спеціальні позначення для попереднього коміту поточної гілки, та кожного з комітів, перед ними:
`HEAD` – поточний останній коміт
`HEAD~` – попередній коміт
`HEAD~~` або `HEAD~2` – коміт, що відрізняється на дві позиції від поточного
і т.д. Таким чином, щоб переглянути ревізію передостаннього коміту треба скористатися командою

```
$ git show HEAD~
```

щоб переглянути ревізію коміту, що стоїть перед передостаннім

```
$ git show HEAD~2
```

Детальніше...

Хования (stashing)

Хования (stashing) бере чорновий стан вашої робочої директорії — тобто ваші змінені супроводжувані файли та індексовані зміни — та зберігає їх у стеку незавершених змін, які ви можете знову використати будь-коли (і навіть на іншій гілці).
Щоб виконати хования (застешувати) виконайте команду

```
$ git stash
```

щоб переглянути список наявних стешів

```
$ git stash list
```

щоб застосувати останнє збережене приховування

```
$ git stash apply
```

або вкажіть номер стешу, відповідно до списку стешів, що повертає команда `git stash list`

```
$ git stash apply stash@{2}
```

Детальніше...

Скидання змін

Якщо потрібно відмінити зміни для деякого файлу, використовують команду restore

```
$ git restore <file>
```

скине зміни у робочій директорії для файлу <file>. Команда

```
$ git restore --staged <file>
```

видалить зміни над файлом <file> з індексу.

Використовують символ . для того, щоб скинути зміни для всіх файлів доданих у індекс або усіх змінених файлів робочої директорії відповідно

```
$ git restore --staged .
```

```
$ git restore .
```

Рух по історії комітів

Для руху по історії комітів використовуються дві команди reset та checkout.

Команду checkout ми вже використовували для переключення між гілками. Проте її можна використовувати для руху по історії комітів (як ми пам'ятаємо, гілка, це лише вказівник на певний коміт).

```
$ git checkout 8bd0d40
```

де `8bd0d40` перші символи хешу коміту.

Задача команди `checkout` переставити вказівник HEAD на вказану гілку або зазначений коміт за хешем. При цьому всі інші вказівники (гілки) залишаються незмінними.

При цьому, якщо у робочій директорії або індексі є зміни, то

Детальніше...

Скидання (reset)

Команда reset не лише пересуває вказівник HEAD на вказаний коміт, а й впливає гілку (переставляє вказівник), на яку вказує HEAD, а також може зачіпати робочу директорію та індекс. Отже, для прикладу, після виконання команди

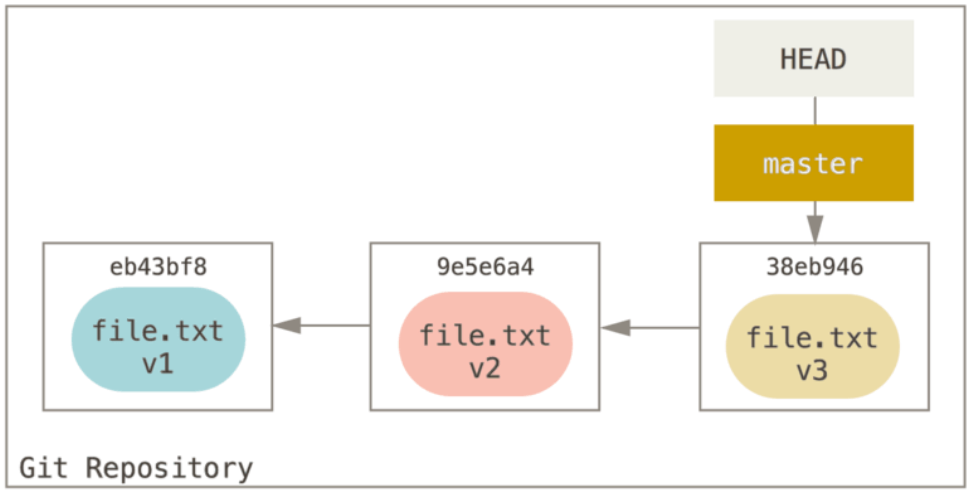
```
$ git reset 8bd0d40
```

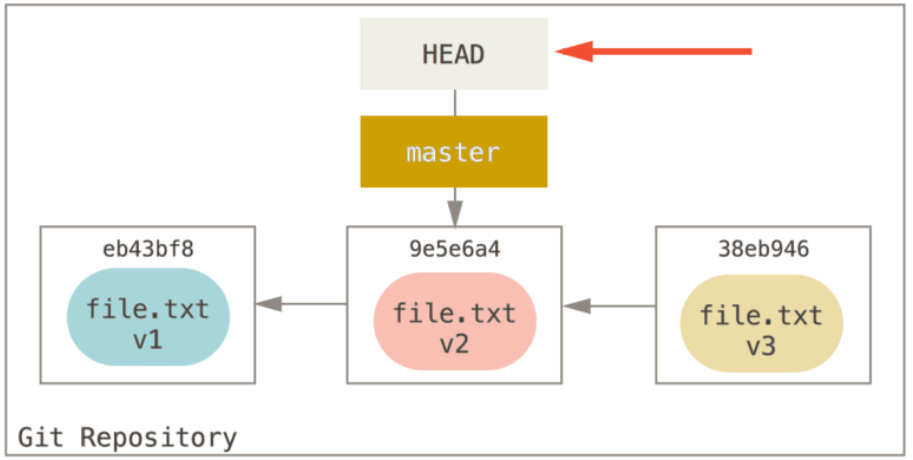
поточна гілка буде вказувати на коміт з хешем `8bd0d40`.

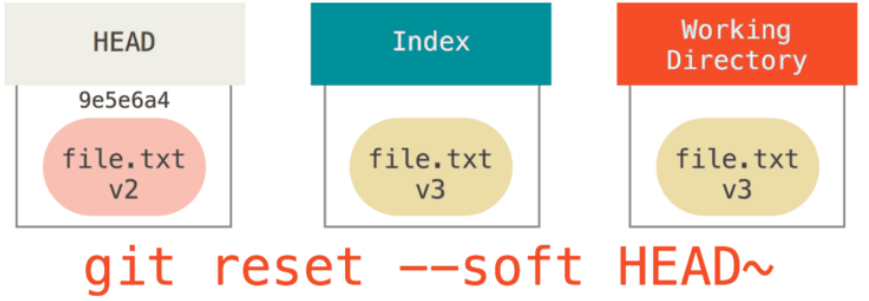
Команда reset (залежно від вказаних додаткових параметрів виконується у три кроки)
Крок 1: перемістити HEAD
якщо команда reset вказується з параметром `--soft`,то наступні кроки не виконуються
Крок 2: оновлення індексу
якщо команда reset вказується з параметром `--mixed`, то наступний крок не виконується
Крок 3: оновлення робочої теки
цей крок виконується, якщо разом з командою reset вказується параметр `--hard`.

Детальніше...

Крок 1: Перемістити HEAD



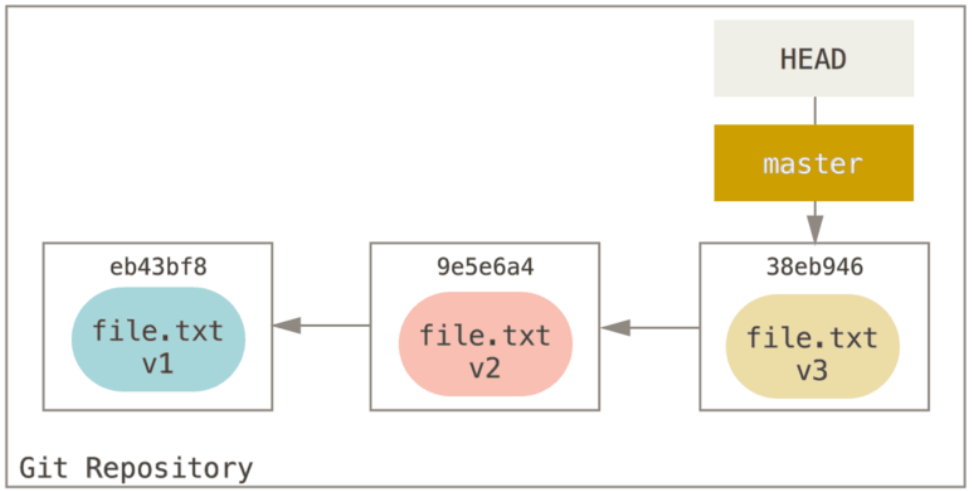


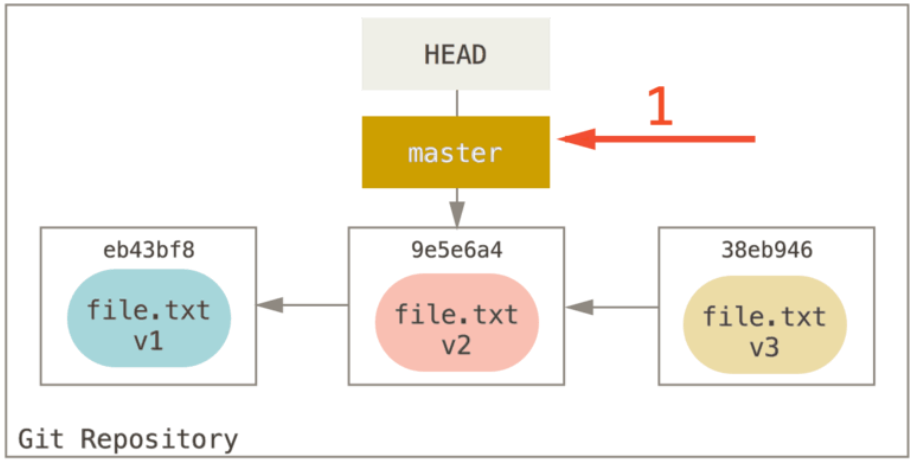


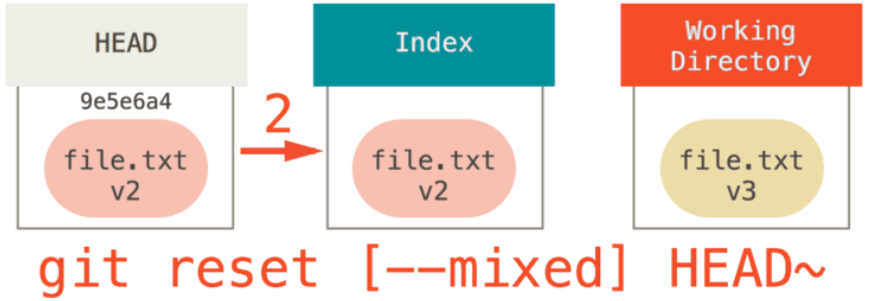

```
$ git reset --soft HEAD~
```

Детальніше...

Крок 2: Оновлення індексу



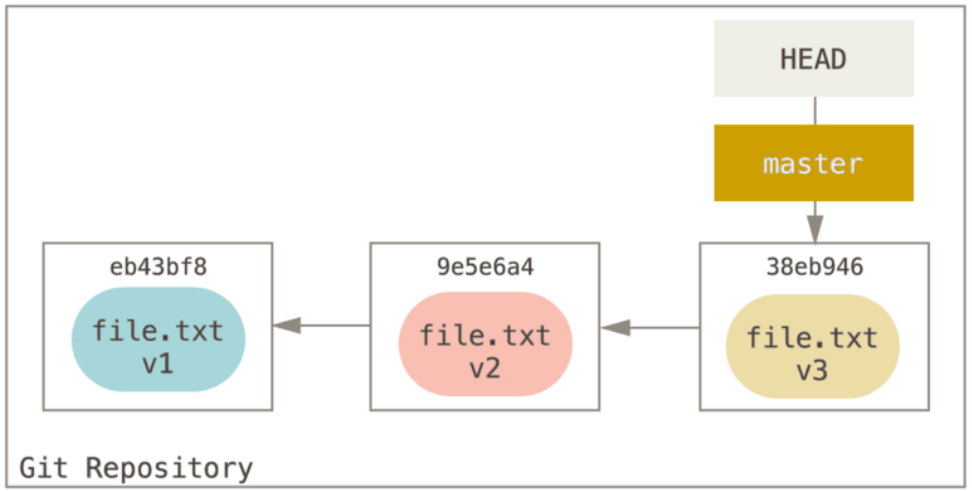


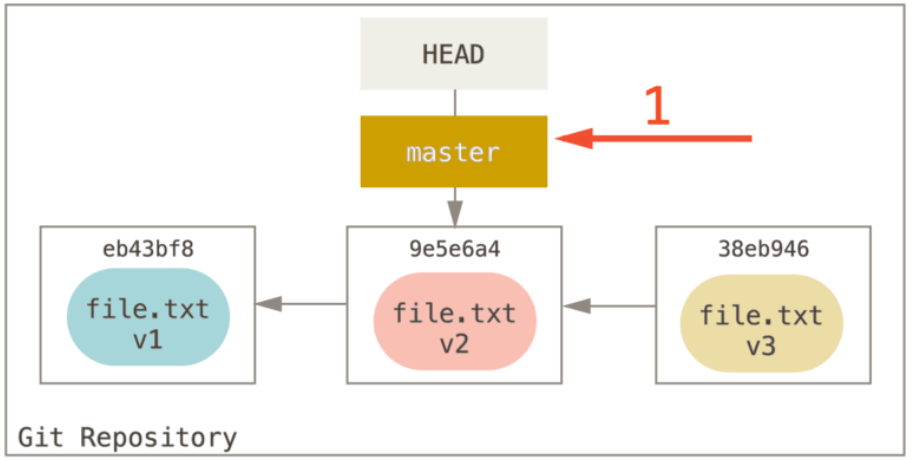


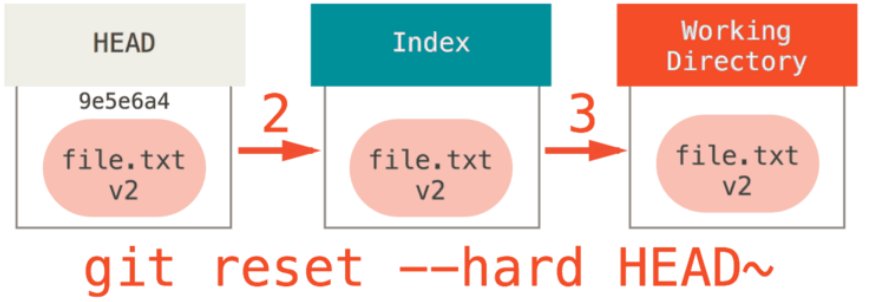

```
$ git reset --mixed HEAD~
```

Детальніше...

Крок 3: Оновлення робочої теки








```
$ git reset --hard HEAD~
```

Детальніше...