Devpractice
Разработка программного обеспечения, технологии и наука

Git для начинающих. Часть 7. Поговорим о HEAD и tree-ish

Автор: Marat Abdrakhmanov | 18.03.2018 2 комментария

В этой статье мы коснемся двух понятий *git*, знание и понимание смысла которых позволит вам более эффективно работать с этой системой контроля версий.

HEAD

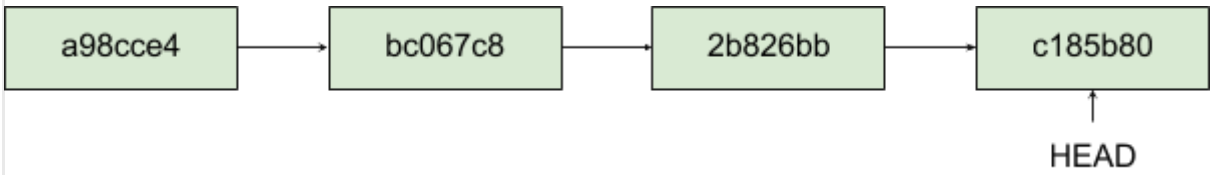
Начнем с *HEAD*. *HEAD* – это указатель, задача которого ссылаться на определенный коммит в репозитории. Суть данного указателя можно попытаться объяснить с разных сторон.

Во-первых, *HEAD* – это указатель на коммит в вашем репозитории, который станет родителем следующего коммита. Для того, чтобы лучше понять это, обратимся к репозиторию, созданному в рамках предыдущей статьи, в этом репозитории сделано шесть коммитов, посмотрим на них.

```
> git log --oneline
cf3d9d8 [add] ignore .tmp files
a7b88ee [create]: git ignore file
c185b80 [create]: header for main
2b826bb [create]: main file of programm
bc067c8 [add]: caption into README file
a98cce4 [create repository]
```

Эти коммиты создавались в порядке от самого нижнего (*a98cce4*) к самому верхнему (*cf3d9d8*). Каждый раз, когда мы отправляли новый коммит в репозиторий, *HEAD* смещался и указывал на него.
Посмотрите на картинку ниже: на ней показана ситуация, когда были отправлены три первых коммита.

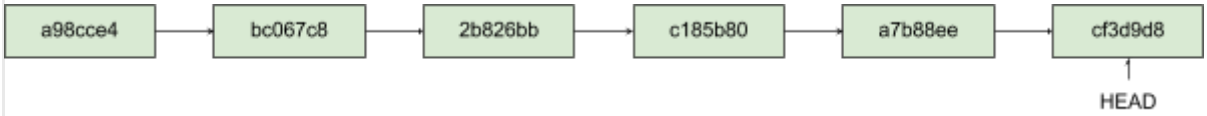
После того как вы отправили коммит с *id = 2b826bb*, указатель *HEAD* стал показывать на него, т.е. данный коммит будет родителем для следующего, и когда мы сделаем еще один коммит, *HEAD* сместится.



Во-вторых, *HEAD* указывает на коммит, относительного которого будет создана рабочая копия во-время операции *checkout*. Другими словами, когда вы переключаетесь с ветки на ветку (о ветвлении в *git* будет рассказано в одной из ближайших статей), используя операцию *checkout*, то в вашем репозитории указатель *HEAD* будет переключаться между последними коммитами выбираемых вами ветвей.

В нашем репозитории пока только одна ветвь – *master*, но и этого будет достаточно, чтобы показать зависимость между положением указателя *HEAD* и операцией *checkout*.

Текущее состояние репозитория выглядит так, как показано на рисунке ниже.



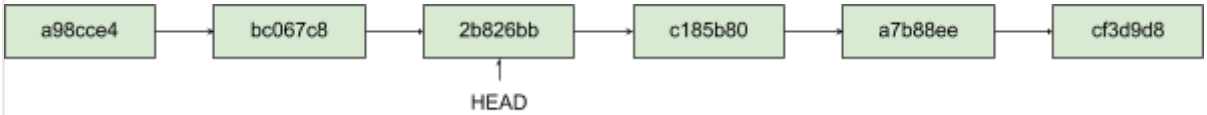
Для того, чтобы скопировать снимок репозитория относительно последнего коммита ветки *master*, т.е. того на который указывает *HEAD*, необходимо выполнить следующую команду.

```
> git checkout master
Switched to branch 'master'
```

Содержимое репозитория, в данном случае, выглядит так.

```
> git log --oneline
cf3d9d8 [add] ignore .tmp files
a7b88ee [create]: git ignore file
c185b80 [create]: header for main
2b826bb [create]: main file of programm
bc067c8 [add]: caption into README file
a98cce4 [create repository]
```

Теперь передвинем указатель *HEAD* на коммит с *id=2b826bb*.



Для этого передадим команде *checkout* идентификатор коммита.

```
> git checkout 2b826bb
Note: checking out '2b826bb'.

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

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 2b826bb... [create]: main file of programm
```

Обратите внимание на текст, который напечатал *git*, после того, как была выполнена эта команда. Нас интересует самая последняя строка "*HEAD is now at 2b826bb…*", теперь *HEAD* указывает на коммит с *id=2b826bb* – именно то, что мы хотели. Посмотрим на текущий список коммитов.

```
> git log --oneline
2b826bb [create]: main file of programm
bc067c8 [add]: caption into README file
a98cce4 [create repository]
```

Git выводит коммиты, которые были сделаны до того коммита, на который ссылается *HEAD*.

Вернем *HEAD* на прежнее место.

```
> git checkout cf3d9d8
Previous HEAD position was 2b826bb... [create]: main file of programm
HEAD is now at cf3d9d8... [add] ignore .tmp files

> git log --oneline
cf3d9d8 [add] ignore .tmp files
a7b88ee [create]: git ignore file
c185b80 [create]: header for main
2b826bb [create]: main file of programm
bc067c8 [add]: caption into README file
a98cce4 [create repository]
```

Все вернулось на прежнее место. Таким образом, вы можете получать в виде рабочей копии содержимое репозитория на момент отправки того или иного коммита. Перейдем в каталог *.git*, в котором находится наш репозиторий, он расположен в корневой директории нашего проекта, и посмотрим его содержимое.

```
> cd .git
> ls -la
total 21
drwxr-xr-x 1 User 197121   0 мар 18 17:10 ./
```

```
drwxr-xr-x 1 User 197121   0 map 18 17:10 ../
-rw-r--r-- 1 User 197121  24 map  5 23:21 COMMIT_EDITMSG
-rw-r--r-- 1 User 197121 184 map  5 23:10 config
-rw-r--r-- 1 User 197121  73 map  5 23:10 description
-rw-r--r-- 1 User 197121  41 map 18 17:10 HEAD
drwxr-xr-x 1 User 197121   0 map  5 23:10 hooks/
-rw-r--r-- 1 User 197121 441 map 18 17:10 index
drwxr-xr-x 1 User 197121   0 map  5 23:10 info/
drwxr-xr-x 1 User 197121   0 map  5 23:10 logs/
drwxr-xr-x 1 User 197121   0 map  5 23:21 objects/
drwxr-xr-x 1 User 197121   0 map  5 23:10 refs/
```

В данном каталоге содержится файл *HEAD*, в нем находится идентификатор, на который ссылается данный указатель. Посмотрим содержимое файла *HEAD*.

```
> cat HEAD
cf3d9d8f7b283267a085986e85cc8f152cca420d
```

HEAD указывает на коммит *cf3d9d8*.

Tree-ish

Понятие *tree-ish* часто используется в документации по *git*. *Tree-ish* – это то, что указывает на коммит, эту сущность мы можем передавать в качестве аргумента для команд *git*. Вот список того, чем может являться *tree-ish*.

Tree-ish	Examples
1. <sha1>	dae86e1950b1277e545cee180551750029cfe735
2. <describeOutput>	v1.7.4.2-679-g3bee7fb
3. <refname>	master, heads/master, refs/heads/master
4. <refname>@{<date>}	master@{yesterday}, HEAD@{5 minutes ago}
5. <refname>@{<n>}	master@{1}
6. @{<n>}	@{1}
7. @{-<n>}	@{-1}
8. <refname>@{upstream}	master@{upstream}, @{u}
9. <rev>^	HEAD^, v1.5.1^0
10. <rev>~<n>	master~3
11. <rev>^{<type>}	v0.99.8^{commit}
12. <rev>^{}	v0.99.8^{}
13. <rev>^{/<text>}	HEAD^{/fix nasty bug}
14. :/<text>	:/fix nasty bug
15. <rev>:<path>	HEAD:README.txt, master:sub-directory/

Рассмотрим работу с *tree-ish* на примере команды *git show*.

```
> git show cf3d9d8f -q
commit cf3d9d8f7b283267a085986e85cc8f152cca420d
Author: Writer  writer@somecompany.c
Date:   Mon Mar 5 23:21:59 2018 +0500
```

```
    [add] ignore .tmp files
> git show -q HEAD
commit cf3d9d8f7b283267a085986e85cc8f152cca420d
Author: Writer <writer@somecompany.com>
Date:   Mon Mar 5 23:21:59 2018 +0500

    [add] ignore .tmp files

> git show -q master
commit cf3d9d8f7b283267a085986e85cc8f152cca420d
Author: Writer <writer@somecompany.com>
Date:   Mon Mar 5 23:21:59 2018 +0500

    [add] ignore .tmp files
> git show -q @{5}
commit cf3d9d8f7b283267a085986e85cc8f152cca420d
Author: Writer <writer@somecompany.com>
Date:   Mon Mar 5 23:21:59 2018 +0500

    [add] ignore .tmp files
```

Во всех примерах, представленных выше, команде *git show* мы передаем различные *tree-ish*, которые на самом деле указывают на одно и тоже место – последний коммит.

> *Отличный курс по git делают ребята из **GeekBrains**, найдите в разделе "Курсы" курс "Git. Быстрый старт", он **бесплатный**!*

<<< Часть 6. Просмотр информации по коммитам

Часть 8. Добавление, удаление и переименование файлов в репозитории>>>

Раздел: Git Git для начинающих Метки: Git, Git для начинающих, Уроки по Git

Git для начинающих. Часть 7. Поговорим о HEAD и tree-ish: 2 комментария

Paul
12.04.2018

А можно чуть подробнее про этот момент?

“Для того, чтобы скопировать снимок репозитория относительно последнего коммита ветки master, т.е. того на который указывает HEAD, необходимо выполнить следующую команду.”
> git checkout master

Я правильно понял, что эта команда “откатывает” предыдущие изменения HEAD?

writer
14.04.2018

Нет, git checkout master скопирует проект из репозитория (ветка master) в рабочую директорию, при этом состояние проекта будет определяться последним коммитом, именно на него обычно указывает HEAD. Т.е. мы получаем снимок репозитория относительно последнего коммита. Надеюсь понятно объяснил))) Если что – пишите!