Devpractice
Разработка программного обеспечения, технологии и наука

Git для начинающих. Часть 6. Просмотр информации по коммитам

Автор: Marat Abdrakhmanov | 07.03.2018 Нет комментариев

Рассмотрим инструмент системы контроля версий *git*, который позволяет делать выборку и представлять пользователю коммиты, отправленные в репозиторий, в соответствии с заданными параметрами.

В прошлом уроке мы разобрались с тем, как фиксировать изменения в рабочей директории и отправлять коммиты в репозиторий. Рабочий процесс с использованием *git,* в упрощенном виде, выглядит следующим образом (пока не рассматриваем работу с удаленным репозиторием):

1. Внесение изменений в рабочую директорию.
2. Отправка изменений в *stage*.
3. Формирование и отправка коммита на базе того, что лежит в *stage*, в репозиторий.

В процессе работы, в вашем репозитории накопится больше количество коммитов и довольно часто будет возникать необходимость их просматривать. *Git* предоставляет удобный способ просмотра информации по коммитам. Для демонстрации возможностей *git*, создадим репозиторий и добавим в него один файл – *README.md*, о том, как это сделать, можете прочитать в предыдущем уроке.

Для просмотра информации по сделанным вами (или вашими коллегами) коммитам используется команда ***git log***.

```
> git log
commit a98cce47b59256d00a853c421af4f7b9f0dc0a29
Author: Writer <writer@somecompany.com>
Date:   Mon Mar 5 23:10:51 2018 +0500

    [create repository]
```

Как видно из полученной информации, в репозиторий был отправлен один коммит с сообщением ***"[create repository]"***, этот коммит сделал пользователь с именем ***Writer***, его *email:* ***writer@somecompany.com***, уникальный идентификатор коммита ***a98cce47b59256d00a853c421af4f7b9f0dc0a29***, и дата и время отправки коммита: ***5 марта 2018 в 23:10:51***.

Внесем еще несколько изменений в наш репозитории. Добавим текст в файл *README.md*.

```
> echo "Project 51" > README.md
```

Зафиксируем эти изменения в репозитории.

```
> git add .
> git commit -m "[add]: caption into README file"
```

Создадим файл *main.c* и добавим его в репозиторий.

```
> touch main.c
> git add .
> git commit -m "[create]: main file of program"
```

Таким образом в нашем репозитории уже должно быть три коммита, проверим это.

```
> git log
commit 2b826bb4929fb1c8166ef05b540ce2cc68f3ebb2
Author: Writer <writer@somecompany.com>
Date:   Mon Mar 5 23:17:08 2018 +0500

    [create]: main file of program

commit bc067c88c427dbedbb02817f9ae25241dcae4d07
Author: Writer <writer@somecompany.com>
Date:   Mon Mar 5 23:15:12 2018 +0500

    [add]: caption into README file

commit a98cce47b59256d00a853c421af4f7b9f0dc0a29
Author: Writer <writer@somecompany.com>
Date:   Mon Mar 5 23:10:51 2018 +0500

    [create repository]
```

Коммиты располагаются от новых к старым. Сделаем ещё несколько изменений.

```
> touch main.h
> git add .
> git commit -m "[create]: header for main"
> touch .gitignore
> git add .
> git commit -m "[create]: git ignore file"
> echo "*.tmp" > .gitignore
> git add .
> git commit -m "[add] ignore .tmp files">
```

Снова получим список всех коммитов.

```
> git log
commit cf3d9d8f7b283267a085986e85cc8f152cca420d
Author: Writer <writer@somecompany.com>
Date:   Mon Mar 5 23:21:59 2018 +0500

    [add] ignore .tmp files

commit a7b88eed6110b6ebb1fc4d96f4399e4cbb8339e7
Author: Writer <writer@somecompany.com>
Date:   Mon Mar 5 23:21:11 2018 +0500

    [create]: git ignore file

commit c185b80ca916af7d6f068450f6cafb073d955c40
Author: Writer <writer@somecompany.com>
Date:   Mon Mar 5 23:20:26 2018 +0500

    [create]: header for main

commit 2b826bb4929fb1c8166ef05b540ce2cc68f3ebb2
Author: Writer <writer@somecompany.com>
Date:   Mon Mar 5 23:17:08 2018 +0500

    [create]: main file of program

commit bc067c88c427dbedbb02817f9ae25241dcae4d07
Author: Writer <writer@somecompany.com>
Date:   Mon Mar 5 23:15:12 2018 +0500

    [add]: caption into README file

commit a98cce47b59256d00a853c421af4f7b9f0dc0a29
Author: Writer <writer@somecompany.com>
Date:   Mon Mar 5 23:10:51 2018 +0500

    [create repository]
```

Количество коммитов в репозитории уже такое, что просматривать информацию о них в том виде, в котором выдает ***git log*** уже неудобно. Для того, чтобы сократить количество показываемой информации можно воспользоваться ключом ***–oneline***, при этом будет выведена часть идентификатора и сообщение коммита.

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

В таком виде работать с коммитами уже намного удобнее. Если вы хотите просмотреть *n* последних коммитов, то укажите количество коммитов после ключа *-n*. Выведем три последних коммита.

```
> git log -n 3 --oneline
cf3d9d8 [add] ignore .tmp files
a7b88ee [create]: git ignore file
c185b80 [create]: header for main
```

Для вывода списка коммитов, начиная с какой-то временной метки, используйте ключ ***–since="<date> <time>"***. Например, получим все коммиты, сделанные после 5-го марта 2018 года 23:21.

```
> git log --since="2018-03-05 23:21:00" --oneline
cf3d9d8 [add] ignore .tmp files
a7b88ee [create]: git ignore file
```

Для вывода списка коммитов до какой-то даты используется ключ ***–until***. Получим список коммитов, сделанных до 5-го марта 2018 года 23:21.

```
> git log --until="2018-03-05 23:21:00" --oneline
c185b80 [create]: header for main
2b826bb [create]: main file of program
bc067c8 [add]: caption into README file
a98cce4 [create repository]
```

Еще одним полезным ключом является ***–author***, который позволяет вывести список коммитов, сделанных конкретным автором.

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

В приведенном выше примере, мы вывели все коммиты сделанные пользователем с именем *Writer*. Т.к. в нашем репозитории все коммиты сделаны от имени данного автора, то при любых других именах, передаваемых параметру *–author*, мы будем получать пустой список.

И, напоследок, рассмотрим еще один инструмент. Если вы работали с *Linux*, то наверное, сталкивались с такой программой как *grep* – это утилита командной строки, которая, в переданном ей тексте, находит вхождения, соответствующие заданному регулярному выражению. Выведем все коммиты, в которых встречается слово *create*.

```
> git log --grep="create" --oneline
a7b88ee [create]: git ignore file
c185b80 [create]: header for main
2b826bb [create]: main file of program
a98cce4 [create repository]
```

Теперь коммиты со словом *add*.

```
> git log --grep="add" --oneline
cf3d9d8 [add] ignore .tmp files
bc067c8 [add]: caption into README file
```

Для более продуктивного использования данной команды рекомендуем ознакомиться с возможностями утилиты *grep*. На этом мы закончим обзор команды *git log*.

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

<<< Часть 5. Создание репозитория и первый коммит Часть 7. Поговорим о HEAD и tree-ish>>>

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