GeekBrains

Git. Базовый курс

Урок 4

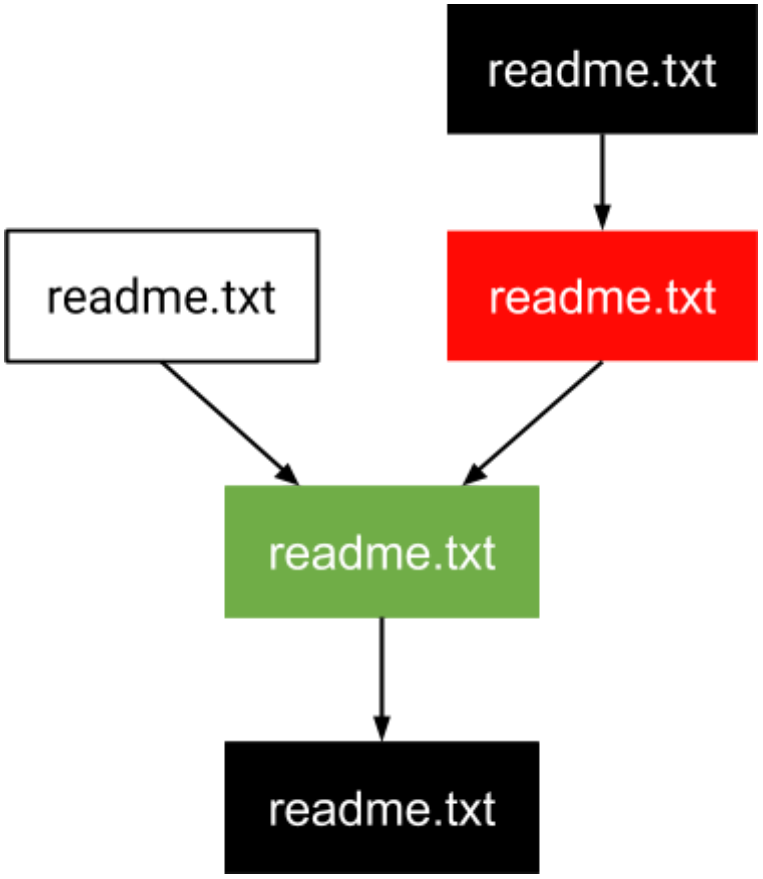
Базовые операции

Добавление файлов в git-проект

Еще раз остановимся на том, как файлы добавляются в git. Пусть есть проект, в нем файлы могут быть в нескольких состояниях:

- отслеживаемом,
- не отслеживаемом.

В отслеживаемом состоянии файл и его содержимое зарегистрированы в git, в неотслеживаемом — файл новый и git о нем пока ничего не знает.



В любом случае git реагирует на оба типа файлов и помечает их как доступные для добавления. Такие файлы могут быть переведены в индексированное состояние при помощи команды **git add**. В будущий коммит войдут только те файлы, которые находятся в индексе.

Можно добавлять или убирать файлы из индекса. После выполнения команды **git commit** содержимое индекса превращается в снимок системы контроля версий, а содержимое индекса обнуляется.

Давайте попробуем пройти весь цикл. Пусть в проекте имеется один единственный файл hello.txt. Для того, чтобы выяснить статус проекта, мы можем воспользоваться командой **git status**:

```
$ git status
```

```
On branch master
nothing to commit, working tree clean
```

Сейчас команда сообщает нам, что изменений в проекте нет. Заведем еще один файл — readme.txt:

```
$ echo 'Н о в ы й  п р о е к т' > readme.txt
```

Здесь мы воспользовались командой **echo**, но для создания такого файла можно использовать любой удобный вам способ, например, текстовый редактор.

Проверяем статус git-проекта при помощи команды **git status**:

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    readme.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Git обнаруживает новый файл readme.txt, и выводит его в отчете красным цветом, т. е. файл помечен как неотслеживаемый.

Файл hello.txt у нас отслеживается, модифицируем и его:

```
$ echo 'Н о в а я  с т р о к а' >> hello.txt
```

Повторно запрашиваем статус проекта:

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   hello.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
```

```
    readme.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Файл hello.txt также помечается красным цветом, но размещается в отдельном списке. Файл hello.txt — отслеживаемый файл, поэтому он отмечен как измененный modified.

Вывод команды **git status** довольно объемный, можно добиться более компактного вывода, если добавить ей параметр **-s**:

```
$ git status -s
 M hello.txt
?? readme.txt
```

Новые, неотслеживаемые файлы помечены **??** слева от них, файлы, добавленные в отслеживаемые, помечены A, отредактированные файлы помечены M и так далее.

Git анализирует только отслеживаемые файлы, поэтому, если мы захотим выяснить, какие изменения были внесены в кодовую базу проекта, он выведет изменения только для hello.txt. Воспользуемся командой **git diff**, чтобы в этом убедиться:

```
$ git diff
diff --git a/hello.txt b/hello.txt
index af5626b..b356b4a 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1 +1,2 @@
 Hello, world!
+Новая строка
```

Итак, мы получаем изменения по файлу hello.txt, про файл readme.txt git не упоминает, так как он неотслеживаемый.

Чтобы файл начал отслеживаться, его необходимо добавить в индекс, делается это при помощи команды **git add**:

```
$ git add .
```

Мы могли бы два раза использовать команду для **git add**, чтобы добавить файл hello.txt и readme.txt. Вместо этого используем символ точки, чтобы добавить оба файла за один раз. Запрашиваем статус git-проекта:

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   hello.txt
    new file:   readme.txt
```

Оба файла выводятся зеленым цветом, это значит, что они добавлены в индекс. Если сейчас сделать коммит, снимок системы, изменения в обоих файлах попадают в этот коммит.

Как же узнать, что падает в коммит?

Если мы сейчас выполним команду **git diff**, она ничего не вернет:

```
$ git diff
```

В проекте нет неотслеживаемых файлов, нет изменений вне индекса. Однако, если мы добавим в команду параметр **--cached**, мы можем увидеть изменения, которые сейчас находятся в индексе:

```
$ git diff --cached
diff --git a/hello.txt b/hello.txt
index af5626b..b356b4a 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1 +1,2 @@
 Hello, world!
+Новая строка
diff --git a/readme.txt b/readme.txt
new file mode 100644
index 0000000..9c68326
--- /dev/null
+++ b/readme.txt
@@ -0,0 +1 @@
+Новый проект
```

Итак мы можем посмотреть какие файлы войдут в будущий коммит и какие изменения будут внесены в каждый из файлов. Зеленые строки с плюсом показывают новые строки, красные строки с минусом — убранные. Добьемся, чтобы в файле hello.txt пропала строка Hello, world!

Для этого перезапишем файл hello.txt строкой 'Новая строка':

```
$ echo 'Новая строка' > hello.txt
```

Давайте запросим статус:

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   hello.txt
    new file:   readme.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   hello.txt
```

Обратите внимание, что файл hello.txt имеется и в зеленом списке индекса, и в красном списке незафиксированных изменений. Если сейчас выполнить команду git commit, изменения в зеленом списке попадут в коммит, а в красном — нет. Мы хотим, чтобы изменения попали в индекс, поэтому выполняем команду **git add**:

```
$ git add .
```

Запрашиваем статус:

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   hello.txt
    new file:   readme.txt
```

Итак, все изменения в индексе. Давайте еще раз посмотрим, что именно изменено в файле:

```
$ git diff --cached
diff --git a/hello.txt b/hello.txt
index af5626b..94c1a48 100644
--- a/hello.txt
```

```
+++ b/hello.txt
@@ -1 +1 @@
-Hello, world!
+Новая строка
diff --git a/readme.txt b/readme.txt
new file mode 100644
index 0000000..9c68326
--- /dev/null
+++ b/readme.txt
@@ -0,0 +1 @@
+Новый проект
```

В файле hello.txt строка заменяется новой, а файл readme.txt сам по себе новый. Зафиксируем изменения, выполняем команду **git commit**:

```
$ git commit -m 'Добавляем файл readme.txt'
[master e0c800c] Добавляем файл readme.txt
 2 files changed, 2 insertions(+), 1 deletion(-)
 create mode 100644 readme.txt
```

Теперь, если запросить статус проекта, можно убедиться, что новых изменений в проекте нет:

```
$ git status
On branch master
nothing to commit, working tree clean
```

Команда **git add .** настолько часто предваряет **git commit**, что последняя команда снабжается дополнительным параметром **-a**. Этот параметр автоматически выполняет перевод всех измененных файлов в индексное состояние.

Добавим в каждый из наших файлов по одной строке:

```
$ echo 'Другая строка' >> hello.txt
$ echo 'Другая строка' >> readme.txt
```

Запрашиваем состояние проекта:

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
```

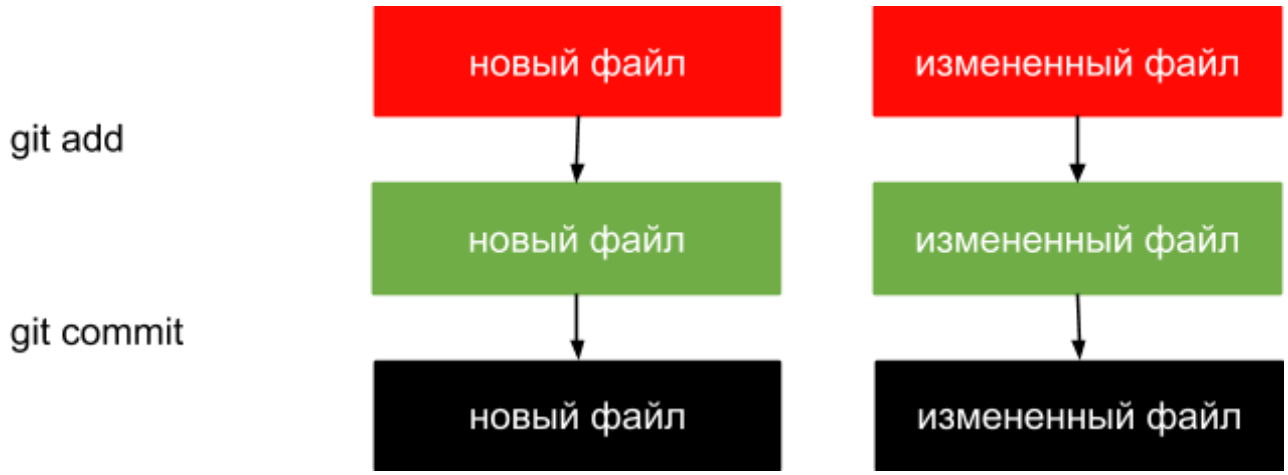
```
    modified:   hello.txt
    modified:   readme.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

```
$ git diff
diff --git a/hello.txt b/hello.txt
index 94c1a48..33637bc 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1 +1,2 @@
 Новая строка
+Другая строка
diff --git a/readme.txt b/readme.txt
index 9c68326..91bd8b0 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1 +1,2 @@
 Новый проект
+Другая строка
```

Можем сразу добавить все изменения одним вызовом **git commit**:

```
$ git commit -am "Изменения в README файле"
[master 9992e6d] Изменения в README файле
 2 files changed, 2 insertions(+)
```

Эта команда эквивалентна последовательному вызову **git add** и **git commit**:



Итак, если у нас есть новый неотслеживаемый файл или измененный отслеживаемый, в первую очередь мы должны перевести их в индексированное состояние при помощи команды **git add**. Все, что находится в индексе, можно превратить в снимок проекта, в коммит, при помощи команды **git commit**.

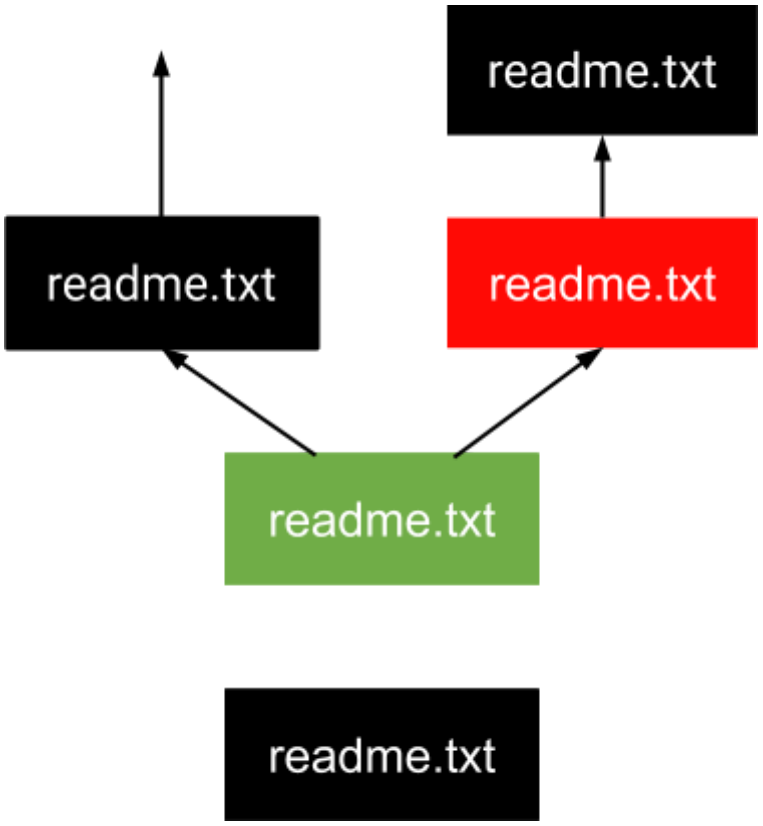
Удаление файлов

Что делать, если файл добавлен по ошибке и его необходимо удалить? Для этого в git тоже есть команды.

Однако прежде, чем мы отправимся в консоль, давайте вспомним, что файлы могут быть в нескольких состояниях:

- неотслеживаемые — это новые файлы, о которых git ничего не знает;
- отслеживаемые — это старые измененные файлы, git знает их предыдущие состояния и в любой момент готов выдать вам ранее сохраненный вариант.

Кроме того, мы можем успеть перевести файлы в индекс и тут мы тоже можем ошибиться: из всех этих трех состояний файл можно удалить. Черные прямоугольники на рисунке ниже обозначают предыдущий и будущий коммит — это зафиксированные состояния. Удалить данные оттуда не так просто.



Это возможно, но на текущем этапе лучше считать, что все, что туда попало, зафиксировано и удалению не подлежит. Давайте воспроизведем ситуацию, когда нам необходимо откатить изменения.

Добавим новый файл to_delete.txt:

```
$ echo 'Ой, это нужно удалить' > to_delete.txt
```

И изменим существующий hello.txt:

```
$ echo 'Ой, это нужно удалить' >> hello.txt
```

Посмотрим статус проекта:

```
$ git diff
diff --git a/hello.txt b/hello.txt
index 33637bc..6062e0f 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1,2 +1,3 @@
 Новая строка
 Другая строка
+Ой, это нужно удалить
```

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   hello.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    to_delete.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Про неотслеживаемый файл git ничего не знает, поэтому, чтобы от него избавиться, достаточно его просто удалить:

```
$ unlink to_delete.txt
```

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   hello.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Для этого не нужны средства git, удаляем файл любым удобным способом. Например, при помощи консольной утилиты unlink. Подойдет и любой другой способ.

С файлом hello.txt так уже не получится, он отслеживаемый. Кроме того, мы не хотим удалять файл, мы хотим получить последнее сохраненное состояние в отчете, и git нам уже подсказывает что делать дальше с файлом:

- либо индексировать при помощи команды **git add**,
- либо откатывать изменения при помощи команды **git checkout --**.

Давайте откатим изменения:

```
$ git checkout -- hello.txt
$ git status
On branch master
nothing to commit, working tree clean
```

Как видим, изменения были затерты. А что делать, если файл уже добавлен в индекс? Давайте смоделируем ситуацию:

```
$ echo 'О й ,  э т о  н у ж н о  у д а л и т ь' >> hello.txt
```

Индексируем файл:

```
$ git add hello.txt
```

Запрашиваем статус git-проекта:

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
```

```
    modified:   hello.txt
```

При помощи команды **git diff** можно узнать, что планируется для добавления в будущий коммит:

```
$ git diff --cached
diff --git a/hello.txt b/hello.txt
index 33637bc..6062e0f 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1,2 +1,3 @@
 Новая строка
 Другая строка
+Ой, это нужно удалить
```

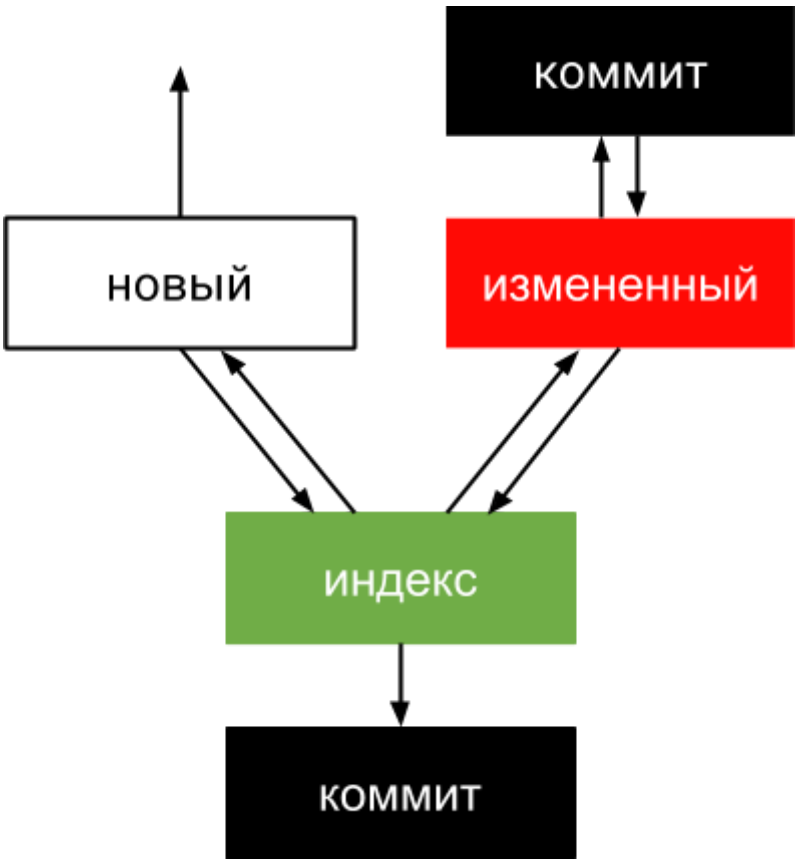
Здесь git нам тоже приходит на помощь: в отчете команды git status приводится подсказка, как отменить проиндексированное состояние при помощи команды **git reset HEAD**:

```
$ git reset HEAD hello.txt
Unstaged changes after reset:
M    hello.txt
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   hello.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Мы переводим файл в неиндексируемое состояние, если мы хотим откатиться к последнему коммиту, мы уже знаем, что делать:

```
$ git checkout -- hello.txt
$ git status
On branch master
nothing to commit, working tree clean
```

Мы откатились к последнему коммиту:



Какие бы изменения ни производились над проектом, вы всегда можете либо довести их до нового снимка состояния, коммита, либо откатиться назад до последнего стабильного состояния.

Используемые источники

Для подготовки методического пособия мы использовали эти ресурсы:

- Официальная документация Git **https://git-scm.com/book/ru/v2**.