GeekBrains

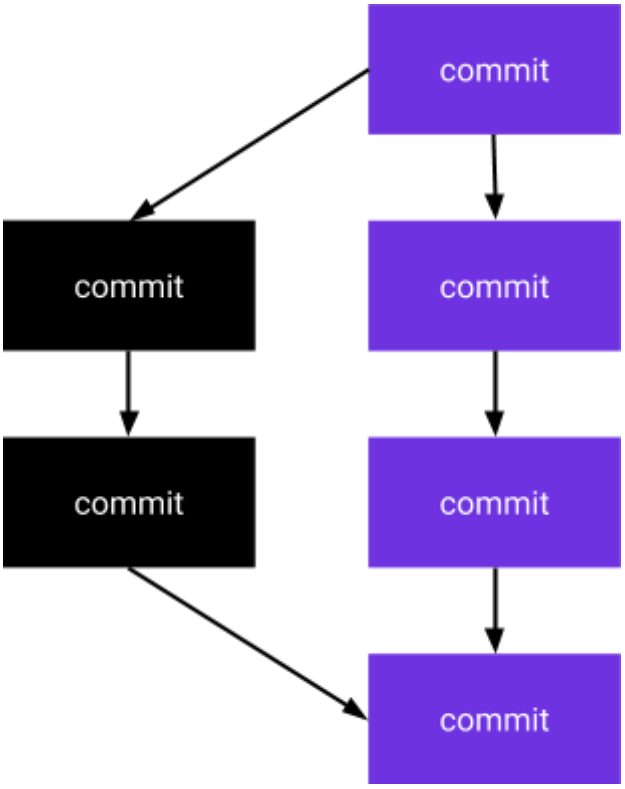
Git. Базовый курс

Урок 6

Ветки

Ветки

Система контроля версий мало бы отличалась от системы резервного копирования, если бы позволяла только делать снимки состояния проекта. Особенностью таких систем является возможность создания нескольких веток, которые, в свою очередь, являются основой для командной работы над проектом.



Сразу после создания проекта появляется основная ветка master:

```
$ git branch
* master
```

В этом можно убедиться при помощи команды **git branch**. Данная команда выводит список доступных веток, помечая текущую звездочкой: пока у нас только одна ветка master, но скоро мы это исправим.

Создание ветки

Если после ключевого слова **branch** указать название, будет создана новая ветка:

```
$ git branch feature
$ git branch
```

Для переключения на новую ветку можно воспользоваться командой **git checkout**:

```
$ git checkout feature
$ git branch
  feature
* master
```

Процесс создания новой ветки можно сократить до одной команды **git checkout**, если воспользоваться ключом **-b**:

```
$ git checkout -b test
Switched to a new branch 'test'
$ git branch
  feature
  master
* test
```

Переименование ветки

Переименовать ветку можно при помощи команды **git branch**, для этого ей необходимо передать параметр **-m**, после которого указывается имя ветки и ее новое имя.

```
$ git branch -m feature to_delete
$ git branch
  master
* test
  to_delete
```

Мы только что переименовали ветку **feature** в **to_delete**. Для удаления ветки следует воспользоваться параметром **-D**:

```
$ git branch -D to_delete
Deleted branch to_delete (was beefeb1).
$ git branch
  master
* test
```

Как видим, **to_delete** из списка веток пропала. После создания и переключения новые коммиты будут размещаться в текущей ветке, не затрагивая историю изменений основной ветки **master**.

Слияние веток

Давайте создадим в проекте новый файл branch.txt и запросим статус проекта:

```
$ touch branch.txt
$ git status
On branch test
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    branch.txt

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

Команда **git status** в первой строке отчета сообщает о том, что мы находимся на ветке test, чтобы мы случайно не внесли незапланированные изменения в другую ветку. Далее сообщается, что обнаружен новый файл branch.txt.

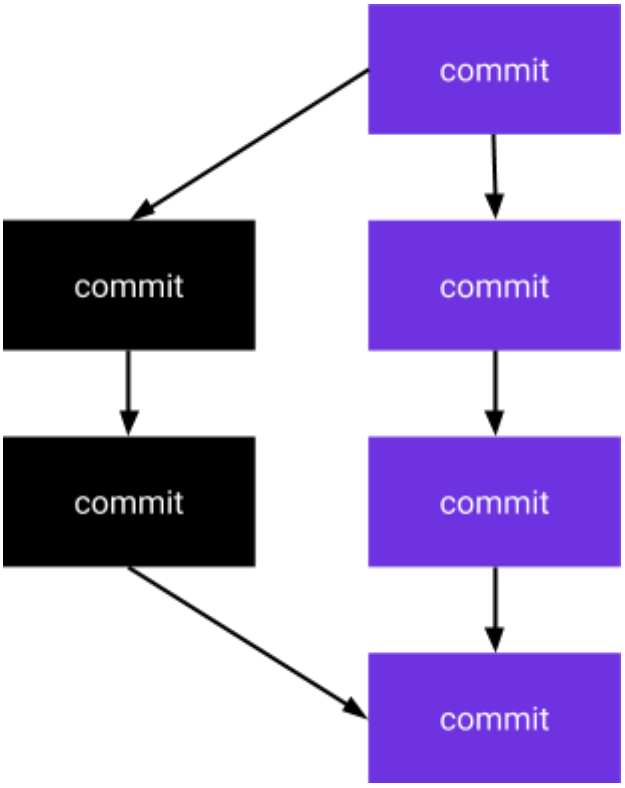
Добавим изменения в новый коммит:

```
$ git add .
$ git commit -m "Новый файл branch.txt"
[test eb19cc9] Новый файл branch.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 branch.txt
```

При помощи команды **git log** запросим историю проекта:

```
$ git log --pretty=format:"%h -%d %s %cd" --graph
* eb19cc9 - (HEAD -> test) Новый файл branch.txt Sun Sep 8 22:25:55 2019 +0300
* beefeb1 - (master) Adding gitignore Sat Mar 30 12:18:10 2019 +0300
* fd076e9 - Изменения в файлах Sun Mar 24 13:52:41 2019 +0300
* c5e5079 - Добавляем файл readme.txt Sun Mar 24 13:50:32 2019 +0300
* 08c3469 - Новый комментарий Sun Mar 24 12:34:51 2019 +0300
* 61c926a - Еще один комментарий Sun Mar 24 12:26:38 2019 +0300
* e6e3fd1 - Comment Sun Mar 24 12:24:07 2019 +0300
* 09056ae - Первый коммит Sun Mar 24 12:21:55 2019 +0300
```

Новый коммит находится в ветке test, ветка master его не содержит.



Коммиты одной ветки можно добавить в другую ветку, такая процедура называется слиянием. Выполнить слияние можно при помощи команды **git merge**.

В момент слияния мы должны находиться в ветке, куда будем сливать изменения, поэтому переключаемся в ветку master:

```
$ git checkout master
```

Давайте, чтобы было интереснее, создадим коммит в ветке master. Для этого создадим новый файл master.txt и сформируем коммит:

```
$ touch master.txt
$ git add .
$ git commit -m "Новый файл master.txt"
[master 66096a3] Новый файл master.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 master.txt
```

Смержим изменения из ветки test в master:

```
$ git merge test
Merge made by the 'recursive' strategy.
 branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
```

```
 create mode 100644 branch.txt
```

Если мы сейчас выполним команды **git log** с параметром **graph**, можем увидеть слияние веток, отрисованное псевдографикой:

```
$ git log --pretty=format:"%h -%d %s %cd" --graph
*   62acb20 - (HEAD -> master) Merge branch 'test' Sun Sep 8 22:27:19 2019 +0300
|\
| * eb19cc9 - (test) Новый файл branch.txt Sun Sep 8 22:25:55 2019 +0300
* | 66096a3 - Новый файл master.txt Sun Sep 8 22:27:03 2019 +0300
|/
* beefeb1 - Adding gitignore Sat Mar 30 12:18:10 2019 +0300
* fd076e9 - Изменения в файлах Sun Mar 24 13:52:41 2019 +0300
* c5e5079 - Добавляем файл readme.txt Sun Mar 24 13:50:32 2019 +0300
* 08c3469 - Новый комментарий Sun Mar 24 12:34:51 2019 +0300
* 61c926a - Еще один комментарий Sun Mar 24 12:26:38 2019 +0300
* e6e3fd1 - Comment Sun Mar 24 12:24:07 2019 +0300
* 09056ae - Первый коммит Sun Mar 24 12:21:55 2019 +0300
```

Конечно, отслеживать параллельные ветки и их слияние удобнее в графическом клиенте.

Псевдонимы команд

Для объемных команд можно использовать короткие запоминающиеся псевдонимы. Создать псевдоним можно при помощи команды git config, добавив свойство alias, после него следует указать имя псевдонима.

Давайте введем псевдоним gg для команды **git log** с параметром **--graph**:

```
$ git config --global alias.gg 'log --pretty=format:"%C(yellow)%h%Creset
-%C(green)%d%Creset %s %C(bold blue)(%cn) %C(green)%cd%Creset" --graph
--date=format:"%d.%m.%Y %H:%M"'
```

Элементы в лог-выводе мы подсвечиваем разными цветами. Теперь можем воспользоваться командой:

```
$ git gg
*   62acb20 - (HEAD -> master) Merge branch 'test' (Igor Simdyanov) 08.09.2019 22:27
|\
| * eb19cc9 - (test) Новый файл branch.txt (Igor Simdyanov) 08.09.2019 22:25
* | 66096a3 - Новый файл master.txt (Igor Simdyanov) 08.09.2019 22:27
|/
* beefeb1 - Adding gitignore (Igor Simdyanov) 30.03.2019 12:18
* fd076e9 - Изменения в файлах (Igor Simdyanov) 24.03.2019 13:52
```

```
* c5e5079 - Добавляем файл readme.txt (Igor Simdyanov) 24.03.2019 13:50
* 08c3469 - Новый комментарий (Igor Simdyanov) 24.03.2019 12:34
* 61c926a - Еще один комментарий (Igor Simdyanov) 24.03.2019 12:26
* e6e3fd1 - Comment (Igor Simdyanov) 24.03.2019 12:24
* 09056ae - Первый коммит (Igor Simdyanov) 24.03.2019 12:21
```

Как видно, мы получили короткую и запоминающуюся команду для быстрого получения списка коммитов. Вы можете создавать свои собственные псевдонимы, не обязательно для команды git log.

Разрешение конфликтов

Git прекрасно справляется с объединением изменений в разных файлах и даже в разных частях одного файла. Однако при командной работе неизбежны конфликтные ситуации, когда изменения одного разработчика противоречат тем, которые вносит другой разработчик.

Попробуем смоделировать такую ситуацию в ветках master и test. Для этого создадим в ветке master файл hello.php:

```
<?php
  echo "Hello, world!";
```

Проверим состояние git-проекта при помощи команды **git status**:

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

    hello.php

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

Создаем коммит:

```
$ git add .
$ git commit -m 'Adding hello.txt'
[master 6c8b386] Adding hello.txt
 1 file changed, 3 insertions(+)
 create mode 100644 hello.php
```

Переключимся на ветку test:

```
$ git checkout test
Switched to branch 'test'
```

Создадим тут точно такой же файл hello.php, только добавляем еще одну строку — **session_start()**:

```
<?php
  session_start();
  echo "Hello, world!";
```

Создаем коммит в ветке test:

```
$ git add .
$ git commit -m 'Adding hello.txt'
[test 1397f35] Adding hello.txt
 1 file changed, 4 insertions(+)
 create mode 100644 hello.php
```

Опять переключаемся в ветку master:

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

Сливаем изменения из ветки test в master:

```
$ git merge test
Auto-merging hello.php
CONFLICT (add/add): Merge conflict in hello.php
Automatic merge failed; fix conflicts and then commit the result.
```

У нас возникает конфликтная ситуация — Git не смог автоматически слить изменения. Из отчета команды можно видеть, что конфликтная ситуация возникала в файле hello.php. Давайте отправимся в редактор и разрешим ее вручную:

```
<?php
<<<<<<< HEAD
  echo "Hello, world!";
=======
  session_start();
  echo "Hello, world!";
>>>>>>> test
```

Конфликты обозначаются угловыми скобками, чтобы их было хорошо видно в коде. В верхней части приводится код текущей ветки, master, в нижней — той ветви, которая сливается (в данном случае это ветка test).

Нам нужно выбрать правильный вариант или написать новый, который учтет изменения из обеих частей:

```
<?php
  session_start();
  echo "Hello, world!";
```

Оставляем вариант из ветки test. Конфликт разрешен. Теперь необходимо вернуться в консоль и зафиксировать изменения.

```
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

    both added:     hello.php

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

$ git diff
diff --cc hello.php
index 6f956ec,2b463e8..0000000
--- a/hello.php
+++ b/hello.php
@@@ -1,3 -1,4 +1,3 @@@
  <?php
+   session_start();
       echo "Hello, world!";
--
```

Добавим изменения в индекс:

```
$ git add .
```

И создадим коммит:

```
$ git commit -m 'Fix confict'
[master eea73b3] Fix confict

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

$ git gg
*   eea73b3 - (HEAD -> master) Fix confict (Igor Simdyanov) 08.09.2019 22:32
|\
| * 1397f35 - (test) Adding hello.txt (Igor Simdyanov) 08.09.2019 22:30
* | 6c8b386 - Adding hello.txt (Igor Simdyanov) 08.09.2019 22:29
* |   62acb20 - Merge branch 'test' (Igor Simdyanov) 08.09.2019 22:27
|\ \
| |/
| * eb19cc9 - Новый файл branch.txt (Igor Simdyanov) 08.09.2019 22:25
* | 66096a3 - Новый файл master.txt (Igor Simdyanov) 08.09.2019 22:27
|/
* beefeb1 - Adding gitignore (Igor Simdyanov) 30.03.2019 12:18
* fd076e9 - Изменения в файлах (Igor Simdyanov) 24.03.2019 13:52
* c5e5079 - Добавляем файл readme.txt (Igor Simdyanov) 24.03.2019 13:50
* 08c3469 - Новый комментарий (Igor Simdyanov) 24.03.2019 12:34
* 61c926a - Еще один комментарий (Igor Simdyanov) 24.03.2019 12:26
* e6e3fd1 - Comment (Igor Simdyanov) 24.03.2019 12:24
* 09056ae - Первый коммит (Igor Simdyanov) 24.03.2019 12:21
```

Теперь можно быть уверенными, что учтены изменения из обеих веток.

Используемые источники

Для подготовки методического пособия мы использовали эти ресурсы:

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