GeekBrains

Git. Базовый курс

Урок 13

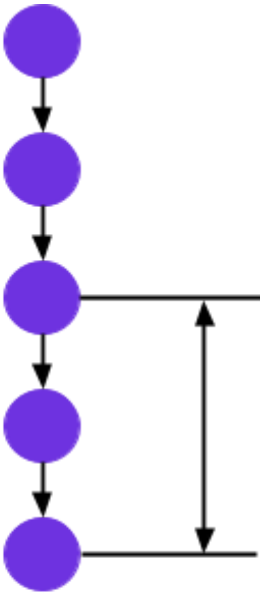
Сдача домашней работы через GitHub

GitHub

Почти все современные компании используют в своей работе Git. Это может быть Git в чистом виде, Git-хостинг GitHub или BitBucket, инсталяция GitLab на собственных мощностях. В любом случае избежать Git в профессиональном коллективе вам не удастся: чем быстрее вы начнете его осваивать, тем вам будет проще на работе. Можно потратить усилия и время на освоение других подсистем и технологий, быстрее двигаться в карьере.

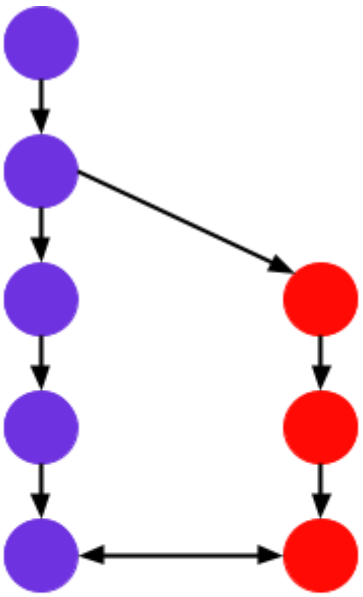
Поэтому на всех курсах GeekBrains, где подразумевается домашняя работа, по возможности используется сдача домашнего задания через git, а точнее, через git-хостинг GitHub.

Если это первая сдача домашней работы или домашние работы не связаны друг с другом, можно просто сослаться на репозиторий.



master

Однако, если домашняя работа предполагает последовательное улучшение программного проекта, хорошо бы предоставлять только ссылку на страницу с изменениями, которые добавились по сравнению с предыдущим домашним заданием. В этом случае можно сравнить два коммита и предоставить ссылку на разницу между ними, либо отпочковать ветку от master, выполнить в новой ветке домашнее заданий, создать pull-реквест в ветку master и предоставить ссылку на него преподавателю.



GitHub автоматически предоставит страницу с изменениями, которые произойдут в случае слияния основной ветки в ветку master.

После того, как домашняя работа принята, ветку задания можно слить в ветку master и приступать к следующему домашнему заданию.

Сдача домашней работы

Давайте начнем с создания собственного GitHub-репозитория. Будем считать что вы уже зарегистрированы ны http://gitbub.com и загрузили свой публичный SSH-ключ в профиль проекта.

Для создания нового репозитория в верхнем меню следует найти значок + и выбрать в выпадающем списке пункт New repository, на странице создания будет предложено назвать проект.

Пусть проект будет называться hello. Для того, чтобы репозиторий был доступен остальным участникам GitHub, его следует оставить открытым. Остальные параметры можно оставить без изменений. Нажимаем кнопку «Создать репозиторий» (Create Repository).

Открывается пустая страница проекта, на которой предлагает два варианта его инициализации:

- Первый вариант предлагает команды для развертывания git-проекта с нуля.
- Второй — подключение уже существующего git-проекта к GitHub.

Команды, которые тут приводятся, должны выполняться на локальной машине. Давайте воспользуемся первым советом и развернем Git-репозиторий с нуля.

Для этого в первую очередь создадим папку проекта, например, hello. Кроме того, давайте создадим какой-нибудь файл, например readme.txt. Git индексирует только файлы, игнорируя папки, поэтому для инициализации проекта, нам потребуется хотя бы один файл.

Первую строчку можно пропустить, так как мы создали файл не при помощи команды echo, а в редакторе. В третьей строчке изменяется имя файла: вместо README.md у нас будет readme.txt. Впрочем, мы можем не указывать имя файла явно, а использовать символ точки для индексации всех новых файлов. Итак, давайте отправимся в консоль и инициализируем git-проект.

Для начала необходимо добраться до папки hello, сделать это можно при помощи команды **cd**:

```
$ cd hello
```

Давайте при помощи команды **pwd** посмотрим текущий путь:

```
$ pwd
```

Инициализируем репозиторий при помощи команды **git init:**

```
$ git init
```

Добавляем файл readme.txt в индекс:

```
$ git add .
```

Создаем первый коммит:

```
$ git commit -m 'П е р в ы й  к о м м е н т а р и й'
```

Теперь нам необходимо при помощи команды **git remote add** добавить адрес удаленного репозитория и закинуть локальные изменения на GitHub при помощи команды **git push**:

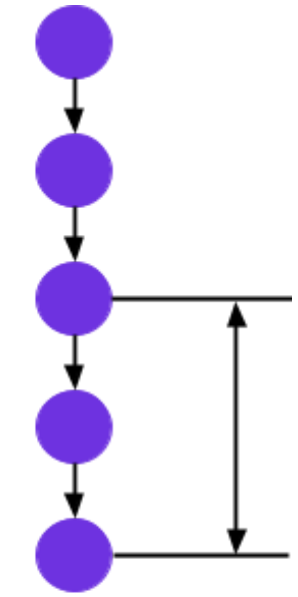
```
$ git remote add origin git@github.com:igorsimdyanov/hello.git
```

Теперь выполняем команду **git push**:

```
$ git push -u origin master
```

Репозиторий инициализирован, давайте убедимся в этом. Если мы сейчас перезагрузим страницу репозитория, то можем увидеть файл hello.txt.

Разница между коммитами



master

Закинем в GitHub несколько коммитов и попробуем получить ссылку на различия между ними. Давайте создадим папку lesson01 и добавим в нее первый файл hello.rb следующего содержания:

```
puts 'Hello, world!'
```

Это небольшая программа на Ruby, которая выводит приветствие в стандартный поток вывода. Давайте оформим ее в виде коммита и забросим на GitHub.

```
$ git status
$ git add .
$ git commit -m 'З а д а н и е 1.1'
```

Отправляем коммит на удаленный github-репозиторий:

```
$ git push origin master
```

Отправляемся в GitHub, чтобы посмотреть, закинут ли файл. У нас появилась папка lesson01, а в ней файл hello.rb.

Давайте создадим еще один файл — each.rb — со следующим содержимым:

```
(1..5).each do |i|
  puts i
end
```

Создаем коммит:

```
$ git add .
$ git commit -m 'Задание 1.2'
$ git push origin master
```

Чтобы получить третий коммит, давайте создадим еще один файл if.rb:

```
if rand(0..1) == 1
  puts 'Попал'
else
  puts 'Промазал'
end
```

Создаем коммит:

```
$ git add .
$ git commit -m 'Задание 1.3'
$ git push origin master
```

Итак, последний из трех запланированных коммитов создан. Переходим в GitHub и проверяем, что все изменения доставлены корректно.

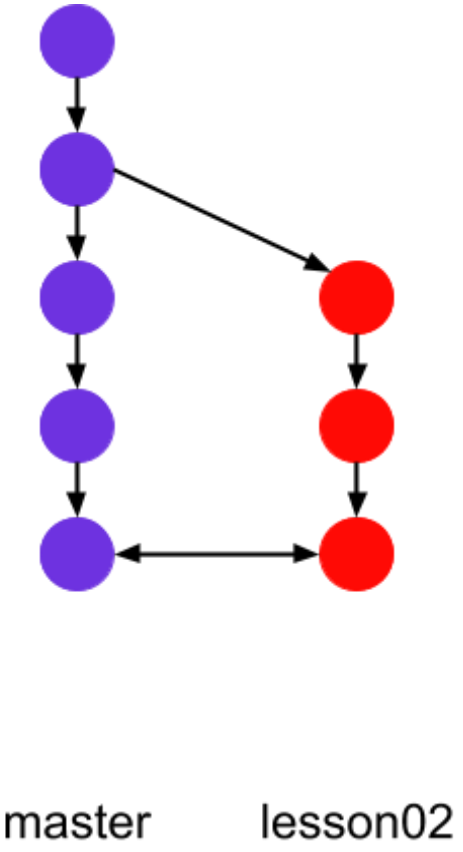
Переходим на основную страницу репозитория, в раздел коммиты. Тут должно быть четыре коммита: один коммит инициации проекта и три других, которые мы добавили в рамках решения первого задания.

Для того, чтобы получить разницу между двумя коммитами, нужно перейти на страницу сравнения. Для этого переходим в Pull Requests и нажимаем кнопку «Новый реквест» https://github.com/your_name/hello/compare (в ссылке вместо your_name следует подставить название вашего git-аккаунта).

Вообще этот инструмент предназначен для сравнения веток, но можно сравнить и коммиты в рамках одной ветки. В первый выпадающий список следует поместить хэш, с которым мы будем сравнивать ветку master (скопировать ее можно со страницы коммитов).

В результате выводится разница между двумя коммитами. При этом ссылку в адресной строке браузера можно отправлять кому-то другому, например, преподавателю. Открыв ссылку, другой разработчик увидит точно такую же страницу, которую видите вы.

Разница между ветками



Однако работать исключительно в ветке master не удобно. Проверка домашней работы может занимать время, в течении которого нельзя будет изменять состояние единственной ветки master.

Гораздо удобнее под каждое из домашних заданий создавать отдельную ветку и отправлять на проверку разницу между веткой задания и веткой master.

Давайте представим, что нам предстоит решение второго домашнего задания. Отпочковываем от master ветку lesson02:

```
$ git checkout -b lesson02
```

Сразу отправляем ее в удаленный репозиторий:

```
$ git push origin lesson02
```

Итак, ветка отправлена, создадим в ней пару коммитов: создаем папку lesson02, а в ней — файл to_a.rb со следующим содержимым:

```
h = { fst: 1, snd: 2 }
p h.to_a
```

Создаем коммит:

```
$ git add .
$ git commit -m 'Задание 2.1'
$ git push origin lesson02
```

На GitHub пока не видно никаких изменений, так как по умолчанию открыта ветка master. Однако, если мы переключимся на ветку lesson02, мы увидим две новые ветки.

Обратите внимание: GitHub сообщает, что недавно была загружена ветка lesson02 и предлагает создать pull-request. Можно нажать на эту кнопку или пойти более традиционным путем, перейдя в раздел pull-request и нажав на ссылку New pull request. Мы опять оказывается на странице сравнения коммитов, только в этот раз нам гораздо проще: у нас есть две ветки для сравнения — master и lesson02, которые мы можем выбрать из выпадающих списков.

Итак. мы получаем diff. В принципе можно послать на проверку его, но лучше завершить создание pull-реквеста, для этого нажимаем на кнопку Create pull request. Создаем пулл-реквест.

Вот эту ссылку уже можно отправить преподавателю на проверку, если у него возникнут замечания, он может вам оставить комментарий.

Их можно поправить, снова сформировать коммит и отправить на GitHub. Если у преподавателя имеются замечания, переписка может быть продолжена. Если он удовлетворен внесенными

изменениями, то нажмет кнопку Resolve conversation. Вы ее тоже можете нажать, если получили ответ «Все нормально».

После того, как вам выставлена оценка, можно слить изменения из ветки lesson02 в master-ветку. Переходим в реквест и нажимаем кнопку Merge pull request. Обратите внимание на стрелочку рядом с кнопкой: вы можете выбрать стратегию слияния, например, если у вас накопилось множество коммитов, вы можете использовать Squash и merge, чтобы предварительно слить все коммиты в один.

Используемые источники

Для подготовки методического пособия мы использовали эти ресурсы:

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