



Фундаментальная и компьютерная лингвистика

РГГУ

Инструкции по Github



А. М. Ивойлова

[github page](#)

Содержание

1	Что такое git и Github	3
2	Что нужно установить	3
2.1	Windows	3
2.2	Linux, MacOS	4
3	Авторизация	5
4	Репозитории	6
5	Работа с чужими репозиториями	9
6	Обновление удаленного репозитория	10
7	Ветки	11
7.1	Одновременная работа в нескольких ветках	13
8	Домашние задания: итог	14

1 Что такое git и Github

Git – это система контроля версий (version control system). Гит нужен для того, чтобы можно было удобно работать над проектами в команде: представьте себе, что вы пишете программу с другом. Допустим, вы написали функцию авторизации пользователя, а ваш друг дописал функцию сохранения логина и пароля; как совместить ваш код? Можно, конечно, выслать друг другу свои версии по почте, а потом склеить в один файл, но это неудобно, а если ваш друг поправил в вашем коде баг или вы переименовали у него какую-то переменную? Вручную выискивать различия тяжело. Для этого и существуют системы контроля версий, на самом деле гит – не единственная (есть еще svn – subversion), но самая популярная.

Github – это соцсеть для программистов. :) На самом деле, Github позволяет командам выкладывать свои проекты в облако и работать над ними вместе, это что-то вроде гугл-доков, за тем исключением, что работаете вы все-таки локально у себя на компьютере, а потом объединяете получившиеся версии в облаке.

И то, и другое работает через терминал (командную строку).

2 Что нужно установить

2.1 Windows

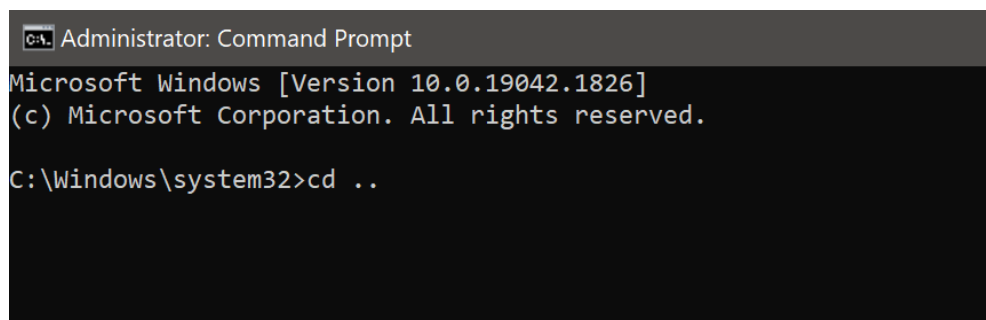
Устанавливаются следующие две вещи:

- [Гит](#)
- [Github CLI](#)

Также вам может понадобиться .NET Framework – это такая штука, без которой не будет работать credentials manager, который хранит все ваши логины-пароли. Он сам скажет об этом при первом запуске, когда потребуется логин-пароль, но можно заранее установить [отсюда](#).

Пользователи Windows – (не)счастливые обладатели командной строки. CLI – консольный интерфейс – это как раз про нее. Вам необходимо уметь ею пользоваться: знать, откуда она запускается, и помнить самые простые команды.

Командную строку легко найти, набрав при открытом меню пуска cmd. Заведите себе привычку всегда запускать ее от имени администратора!



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19042.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd ..
```

Так выглядит командная строка. В ней всегда указан путь: папка, из которой вы работаете. В командной строке можно запускать программы, создавать и удалять папки, просматривать списки файлов в папке и так далее. Всегда обращайтесь внимание на то, какой у вас указан путь.

Чтобы сменить путь, нужно воспользоваться следующими командами:

- `cd ..` – с двумя точками действует точно так же, как стрелка "вверх" в проводнике
- `cd C:\Python` – переместит вас по указанному адресу, в данном случае папка Python на диске C.
- `cd \d D:\Python` – переместит вас на другой жесткий диск (без буковки d на другой диск попасть нельзя)

Вам обычно понадобится открывать командную строку в той папке, в которой, например, лежит ваш проект.

2.2 Linux, MacOS

Если вы счастливый пользователь Linux Ubuntu, у вас уже, скорее всего, предустановлен git. Проверить это можно, набрав в терминале команду

```
git --version
```

В противном случае он устанавливается как

```
sudo apt install git
```

(или apt-get). Если у вас Fedora, то команда будет выглядеть как `sudo dnf install git` (или yum вместо dnf на более старых версиях или на CentOS и Red Hat).

Точно так же устанавливается и Github CLI:

```
sudo apt install gh
```

Пользователям MacOS понадобится [homebrew](#). Дальнейшие действия очень похожи на Линукс:

```
brew install git  
brew install gh
```

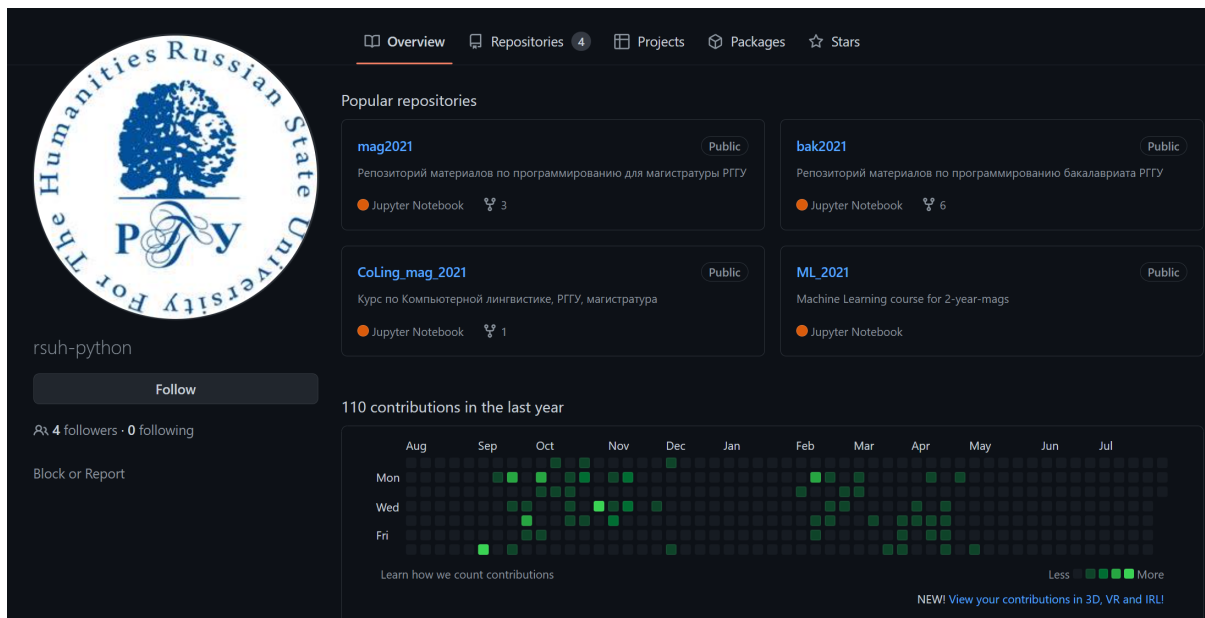
Все эти команды, разумеется, вводятся в терминале: в обеих этих системах он так называется. Он работает примерно так же, как командная строка, и тоже знает команду `cd`, только `\d` будет не нужен, потому что у вас файловая система Unix. В Ubuntu можно открывать терминал в нужной папке через контекстное меню по правому щелчку мыши.

Примечание: если у вас какой-то Линукс и вы не можете найти у себя терминал, скорее всего, вы пользуетесь Endless OS. Советую заменить операционную систему на Ubuntu.

3 Авторизация

Отлично, когда все необходимое установлено, нужно зарегистрироваться собственно на [гитхабе](#). Сам процесс регистрации, полагаю, в коментах не нуждается. Единственное, что нужно здесь с самого начала знать – это как генерировать токены аутентификации.

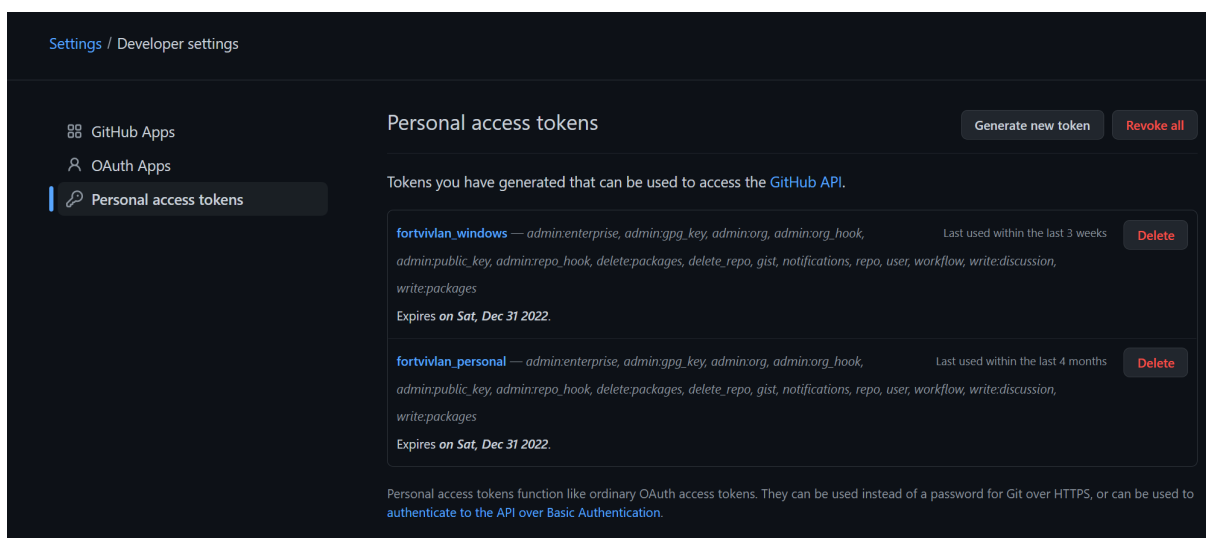
Итак, сам гитхаб выглядит примерно так:



Что здесь есть? Помимо информации о вас, в overview вы видите свои репозитории (что это – поговорим попозже), календарь контрибуций (каждый раз, когда вы что-то добавляете в гитхаб, это отображается), что-то вроде стены с вашими действиями. Во вкладке Stars можно увидеть другие репозитории, которые вы лайкнули – поставили им звездочку. Это удобно для репозиторий, в которые вы хотите время от времени заглядывать.

Все отлично, но пользоваться гитхабом мы с вами будем по большей части из консоли (терминала или командной строки). Вопрос: как авторизоваться в гитхабе оттуда?

Для этого гитхаб обязательно требует (у них такая политика безопасности), чтобы вы сгенерировали токен авторизации (это просто какой-то код вроде пароля). Чтобы его сгенерировать, нужно попасть сюда:



Попасть сюда можно из меню в правом углу экрана (Settings – внизу в колонке вкладок отыщите Developer Settings). Вам нужно будет нажать кнопку Generate new token и выбрать права для этого токена (по умолчанию выбирайте все, вы же для себя генерируете). Потом, когда сайт покажет вам этот токен, обязательно сохраните его куда-нибудь себе! Больше вы этот токен уже не увидите, он вам его не покажет, останется только генерировать новый.

Теперь, когда вы завели аккаунт и сгенерировали токен, нужно настроить git у себя на компьютере.

Вам нужно будет настроить конфигурацию с помощью команды

```
git config --global user.name "ваш логин на гитхабе"
```

и

```
git config --global user.email "ваша почта"
```

Это надо будет сделать только один раз.

Если вы при этом хотите запомнить свой токен и не вводить его каждый раз, то для Windows есть команда

```
git config --global credential.helper manager-core
```

(она у вас спросит все, что надо, и сохранит), а для Linux есть команда

```
git config --global credential.helper store
```

после которой надо будет еще выполнить пустой git pull, чтобы он запросил все, что нужно, и запомнил это (но пользователи StackOverflow не рекомендуют запоминать токен!).

4 Репозитории

Итак, с авторизацией вроде бы разобрались. Теперь о главном!

Что такое репозиторий? Это, по сути, папка с вашим кодом, ваш проект. Допустим, все ваши домашки хранятся в папке prog_hws. Тогда, чтобы выкладывать все домашки на гитхаб (и чтобы гит следил за ними и обновлял версии), нужно сделать эту папку репозиторием. Соответственно, существуют локальные и удаленные (remote) репозитории: те, которые лежат у вас на компьютере, и которые лежат в облаке гитхаба.

Существует минимум два способа завести новый репозиторий.

Первый – создать его через интерфейс сайта и клонировать себе на компьютер. Как это делается:

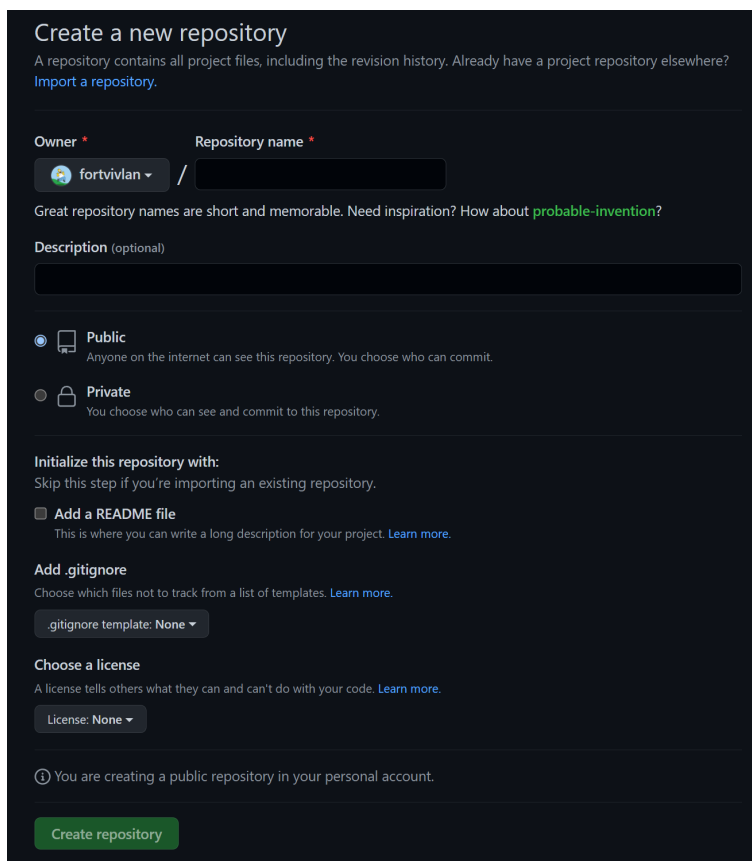
The image shows the 'Create a new repository' page on GitHub. At the top, it says 'Create a new repository' and provides a brief explanation: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.' Below this, there are two main sections. The first section is for repository details, including 'Owner' (a dropdown menu showing 'fortvivlan') and 'Repository name' (a text input field). A note below these fields says 'Great repository names are short and memorable. Need inspiration? How about probable-invention?'. The second section is for repository settings, including 'Description (optional)' (a text area), 'Public' (selected) and 'Private' (unselected) radio buttons, 'Initialize this repository with:' (a section with a checkbox for 'Add a README file'), 'Add .gitignore' (a dropdown menu for '.gitignore template: None'), and 'Choose a license' (a dropdown menu for 'License: None'). At the bottom, there is a green 'Create repository' button and a note: 'You are creating a public repository in your personal account.'

Рис. 1: Создание репо с сайта

1. Заходите во вкладку Repositories, нажимаете кнопку New.
2. Даете репозиторию имя: это то, что будет отображаться в его названии и в адресе.
3. Решаете, будет ли он приватным или публичным.
4. Добавляете Readme (красивее это делать, но можно и не добавлять)
5. Жмете кнопку Create new repository
6. Готово! Теперь у вас откроется страничка вашего созданного репозитория.
7. Чтобы загрузить его себе на компьютер, откройте командную строку в том месте, где вы хотите создать папку с вашим репозиторием, и наберите команду

```
git clone PATH
```

Откуда взять этот адрес? На странице вашего репо нажмите зеленую кнопку Code и скопируйте HTTPS-адрес, который там указан!

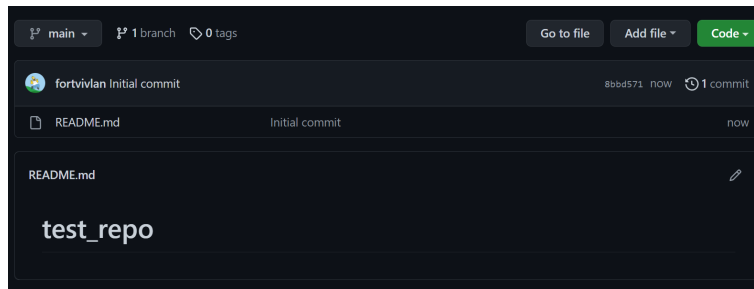


Рис. 2: Страница нового репозитория

Второй способ – завести новый репозиторий с помощью командной строки. Тут уже понадобится `gh`.

1. Создайте локальный репозиторий: подготовьте папку с именем, какое вы хотите, заведите там хотя бы ридми-файл.
2. Откройте командную строку/терминал из этого репозитория и наберите команду

```
git init -b main
```

Эта команда инициализирует ваш репозиторий в гите: он начнет отслеживать версии.

3. Потом выполните команду

```
gh repo create
```

...и отвечайте на вопросы. :) Когда он вас спросит, ответьте, что вы хотите Push an existing local repository to GitHub. После этого останется только выполнить команду `push` с параметрами (про приведенные ниже команды см. дальше):

```
git add -A
git commit -m "initial commit"
git push --set-upstream origin main
```

Таким образом вы заведете главную ветку вашего репозитория, потом уже можно делать обычный `push` без всяких параметров.

Все эти же самые инструкции, только на английском языке и, может быть, чуть более подробно, есть в [документации самого гитхаба](#).

5 Работа с чужими репозиториями

Если вы хотите только скопировать чужой репо себе на компьютер и обновлять его содержимое иногда (а вы будете хотеть это делать с нашими материалами), то достаточно в первый раз скопировать HTTPS-ссылку этого репозитория с его страницы, открыть командную строку/терминал в том месте, куда вы хотите скопировать папку, и выполнить команду

```
git clone PATH
```

Ну и, соответственно, вставить вместо PATH (Ctrl+V, кстати, или Shift+Ins для линукса) адрес, который скопировали.

Эту команду достаточно выполнить только один раз. Для обновления репозитория нужно открывать в его папке командную строку/терминал и набирать команду

```
git pull
```

Будьте бдительны: гиту невдомек, что вы не разработчик – участник проекта, а студент, поэтому если вы измените какой-нибудь файл в репозитории и он же будет изменен автором репозитория, гит будет ругаться и требовать от вас сделать pull request и merge. Чтобы избежать такой неприятности, заведите привычку создавать копии тетрадок с семинарами, ну и я постараюсь файлы лишний раз не менять, а добавлять новые.

Если же все-таки такая проблема случайно произошла, то можно попробовать использовать команду stash (но там не все так просто):

```
git stash
git pull
git stash pop
```

Возможно, также пригодится команда git stash apply (если pop ругается).

Ну а если вы хотите не просто пользоваться результатами чужой работы, а, например, нашли баг у автора и хотите его исправить, нужно репозиторий форкнуть: на его странице нажать кнопку fork. Тогда этот репозиторий скопируется к вам в профиль, и уже оттуда вы можете скопировать его себе локально на компьютер с помощью команды clone и делать с ним, что вам захочется, запустить изменения (об этом см. ниже). Когда вы поправили баг, вы можете кинуть автору оригинального репозитория pull request.

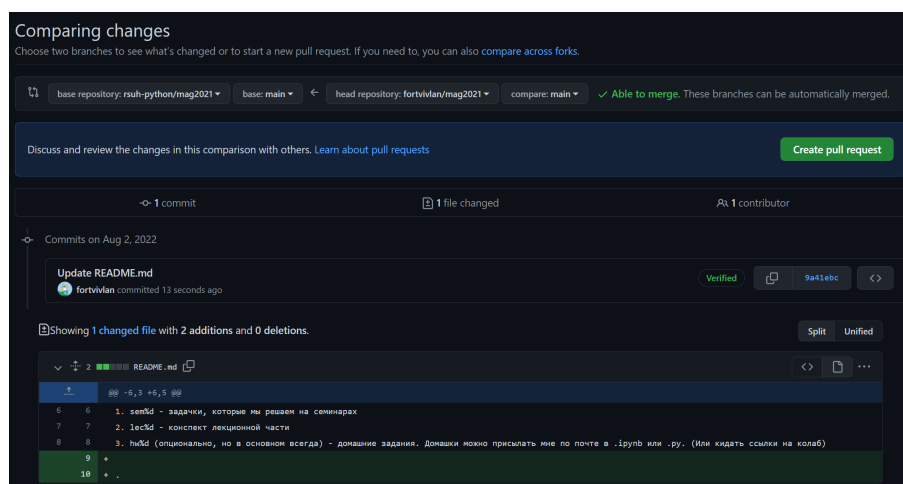


Рис. 3: Пулл-реквест

Чтобы открыть пулл-реквест, нужно, собственно, на странице вашего форка этого репо нажать на кнопку "pull requests" (неожиданно). Там появится сравнение внесенных изменений и оценка того, насколько просто будет склеить версии (на картинке я грустно поставила лишнюю точку в ридми-файл, поэтому, конечно, все просто склеить). Там же, в этом пулл-реквесте, можно обмениваться комментариями.

6 Обновление удаленного репозитория

Допустим, вы создали свой репозиторий и вот хотите сделать свой первый проект. Убедитесь, что в удаленном репозитории не делали никаких изменений (например, не правили ридми-файл), иначе сделайте команду `git pull` для того, чтобы состояние репо было одинаковое. Потом спокойно делайте локально все, что хотите, создавайте в папке репозитория любые файлы, пишите код (в чем угодно, гиту все равно: он, кстати, умеет интегрироваться с некоторыми средами разработки, точно работает с PyCharm, после некоторых танцев с бубном умеет работать с Visual Studio Code). Когда вы решились запустить свой проект, нужно открыть командную строку/терминал из папки с ним и выполнить следующие команды:

```
git add -A
git commit -m "describe commit"
git push
```

Первая из команд говорит гиту добавить в репозиторий указанные файлы (-A означает просто все файлы в папке). Можно указывать какие-то конкретные адреса или, например, выбрать все файлы с расширением `ru`, используя маску: `*.ru`.

Вторая команда добавляет комментарий к коммиту: здесь принято писать, какие изменения вы внесли. Первый коммит обычно называют `initial commit`, дальше принято писать что-нибудь вроде `"add new function for ..."`.

Третья команда – самая главная, она пушит (передает) ваши файлы в удаленный репозиторий, публикует их там. В этом месте гит спросит ваш токен авторизации, если вы его не сохранили, может спросить логин и почту, если вы не настроили `config`.

После этих трех команд, если все выполнилось, можно бежать на сайт смотреть, что получилось. :)

7 Ветки

Если вы работаете над проектом командой, вам может потребоваться одобрение руководителя проекта или коллег для внесения изменений в код. Для того, чтобы эти изменения сделать, но не добавлять их сразу в итоговый вариант, существуют ветки: вы можете сделать версию своего проекта, изменить там все, что нужно, и попросить коллег одобрить. Как это сделать?

Ну, как создать репозиторий, можно почитать выше, а коллабораторы (коллеги-разрабы) приглашаются через меню Settings в вашем репозитории.

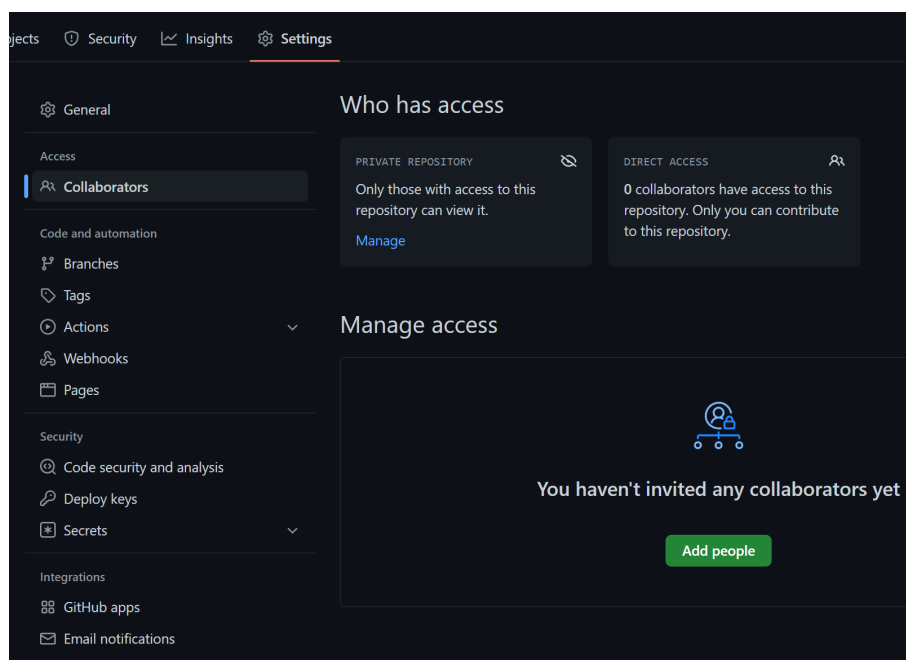


Рис. 4: Добавление друзейшек

Когда вы заходите во вкладку Collaborators, гитхаб из осторожности может спросить у вас пароль – это нормально.

Дальше нужно добавить кого-нибудь. Для ваших домашних заданий вам надо будет добавить в коллабораторы меня.

Вам нужно будет набрать либо почту, либо имя человека, которого вы хотите пригласить. Когда вы добавите его, выбранный вами человек получит письмо с приглашением и может его принять. Вы увидите, что человек принял реквест, когда он в списке коллабораторов будет указан как коллаборатор.

Вот теперь – самое интересное.

Все вместе над одним и тем же проектом разрабы работать не будут: они создадут каждый себе по отдельной ветке, в которой и будут делать то, что хотят. Когда разработчик решит, что он сделал свою работу и готов ее закоммитить в главный проект, он кинет

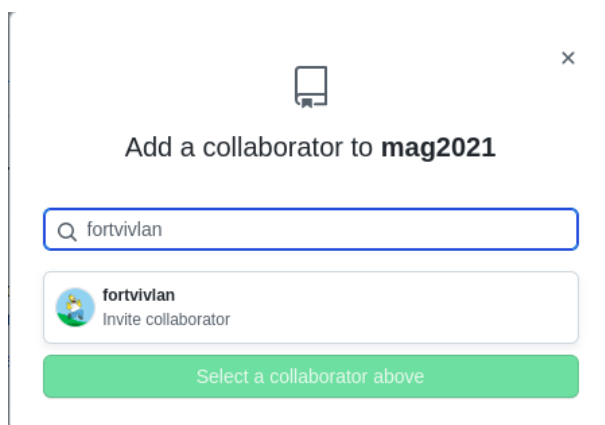


Рис. 5: Как добавить коллаборатора

пулл-реквест и еще может пригласить ревьюера из числа своих коллабораторов: ревьюер одобрит сделанную работу и может сдвинуть ветки (или скажет, что надо поправить).

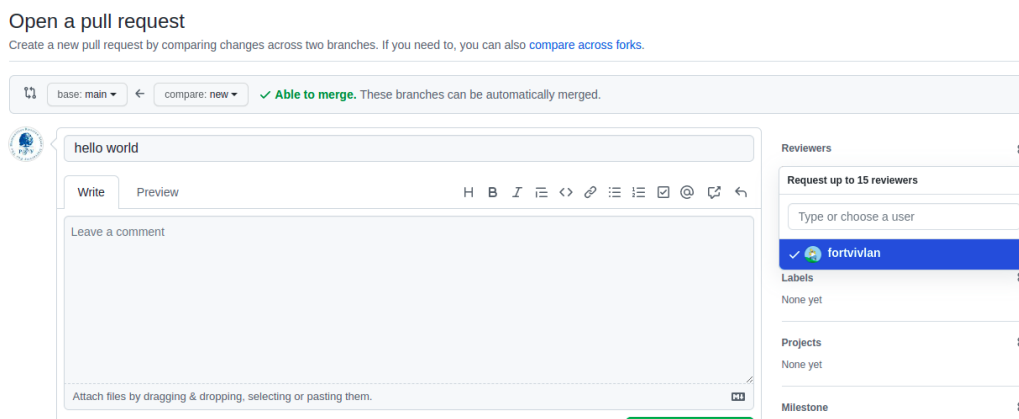


Рис. 6: Выбор ревьюера

Итак, а какими командами пользоваться для работы с бранчами?

Для того, чтобы посмотреть все существующие ветки вашего репозитория, вы можете набрать команду (находясь в его папке):

```
git branch -a
```

Команда выведет список всех существующих веток и зелененьким обозначит ту, в которой вы сейчас работаете. Чтобы создать новую ветку и перейти в нее, нужно ввести команду:

```
git checkout -b NAME
```

И в качестве имени ввести какое-то название для вашей ветки. После того, как вы создали новую ветку, гит автоматически в нее перейдет, и все изменения, которые вы теперь будете делать в репозитории, будут засчитываться в нее: в главной ветке ничего меняться не будет.

Окей, допустим, вы создали ветку hw01, сделали первую домашнюю работу и теперь хотите ее сдать. Процесс публикации новых веток на гитхабе почти такой же:

```
git add -A
git commit -m "add hw01"
git push --set-upstream origin hw01
```

Единственная разница – когда вы только сделали ветку и еще ничего в нее не коммитили, нужно использовать команду `push` с параметрами (вместо `hw01` может стоять любое другое имя, как ваша ветка называется). Если вы захотите что-то запустить еще раз, можно будет уже использовать просто `git push`.

На заметку: что такое `upstream`? Это просто имя другой ветки, с которым будет ассоциироваться ваша ветка (обычно это означает, что у вас есть локально на вашем компьютере ветка `origin`, которая в ее удаленной копии на гитхабе называется `main`).

Отлично, теперь ваша домашняя работа появилась на гитхабе: обратите внимание, чтобы ее увидеть, нужно на странице репозитория переключиться на ветку с ней (там есть меню). Когда вы переключитесь на свою ветку, можно кинуть пулл-реквест и попросить коллаборатора (меня) проверить дз: зайдите в меню `pull requests` и выберите зеленую кнопку `compare and pull request`. В открывшемся окошке можно выбрать ревьюера, как это описывалось выше. Когда вы выберете человека в ревьюеры, он получит письмо о том, что вы кинули пулл-реквест (иначе не получит!). Теперь он может написать вам комментарий (вы получите письмо про комментарий), попросить что-нибудь поправить, а может выбрать `Approve` или даже `Merge`, и тогда ваша ветка склеится с основным репозиторием. Обратите внимание: как коллаборатор, вы сами тоже можете выполнить `Merge`, но не делайте этого, пока не получите `Approve`, а то зачем весь сыр-бор. :)

Чтобы вернуться у себя на компьютере в главную ветку (или переключиться на любую другую), нужно использовать команду:

```
git checkout NAME
```

Например, `git checkout master` – для возвращения в основную ветку. Еще ветки можно удалять через командную строку/терминал локально:

```
git branch -d NAME
```

Вся переписка в пулл-реквестах между тем будет сохраняться, и ее можно пересмотреть в любой момент.

7.1 Одновременная работа в нескольких ветках

Во-первых, необходимо себе уяснить: когда вы делаете новую ветку в гите, гит не копирует все файлы вообще (но отображает их и указывает, какие из них были изменены). При этом, когда вы переключены на эту ветку с помощью команды `checkout`, гит постоянно отслеживает все локальные изменения, поэтому, прежде чем делать домашку, вам нужно переключиться в командной строке на ветку с ней, и только потом уже создавать файл с домашкой и что-то в нем делать.

Если у вас возникает необходимость создать новую ветку и переключиться на нее, когда предыдущая еще не смержена, вы можете это сделать двумя способами:

1. Запустить все изменения в текущей ветке с помощью обычного набора команд (ветка в репозитории обновится), а потом создать новую ветку командой `git checkout -b branchname`.
2. Выполнить команду `git stash --all`, которая сохранит ваши изменения в `stash`: это такое специальное пространство для WIP (work in progress), и тогда уже создавать новую ветку.

Стешей можно делать несколько, посмотреть их можно с помощью команды `show`:

```
git stash show stash@{0}
```

При этом в фигурных скобочках указывается порядковый номер стеша (если вы сделали только один, он будет лежать в 0, если два - то второй можно будет посмотреть по цифре 1).

Стеши автоматически не возвращается, когда вы возвращаетесь на ветку, изменения в которой сохранили с его помощью. Нужно сперва выполнить команду `git add -A` (можно вписывать конкретные названия файлов), а потом команду `git stash apply stash@N`. То есть, например, если у меня была ветка `branch01`, я сделала в ней какие-то изменения, ничего не запустила на гитхаб и захотела переключиться между ветками, я делаю следующее:

```
git stash --all
git checkout branch02
...вношу изменения, делаю или другой стеш...
git checkout branch01
git add -A
git stash apply
...или (если стешей было несколько и мне нужно выбрать не последний по умолчанию, а
git stash apply stash@{1}
```

Как понять, какой стеш ваш? Во-первых, можно попросить гит вывести список всех стешей:

```
git stash list
```

Во-вторых, когда вы просите гит показать конкретный стеш, он выводит, какие в этом стеше были изменения, ну и дальше остается догадаться, какой вам нужен. Гит также пишет, к какой ветке он относится, так что не запутаетесь.

8 Домашние задания: итог

Таким образом, что вам нужно делать, чтобы успешно сдавать домашние задания?

1. Установить нужные приложения
2. Зарегистрироваться на гитхабе

3. Создать репозиторий для своих домашек, приватный или публичный – все равно
 4. Пригласить меня в коллабораторы
 5. Завести новую ветку `hw...`
 6. Переключиться на нее в гите
 7. Делать домашку!
 8. Закоммитить новую ветку
 9. Кинуть пулл-реквест и попросить меня о ревью
 10. Получить фидбек, внести поправки, снова закоммитить...
 11. Домашка смержена – удаляем ненужную ветку и возвращаемся к пункту 5
-