

Основы программной инженерии (ПОИТ)
Технологии разработки программного обеспечения (ИСИТ)

Системы контроля версий. Часть 2

План лекции:

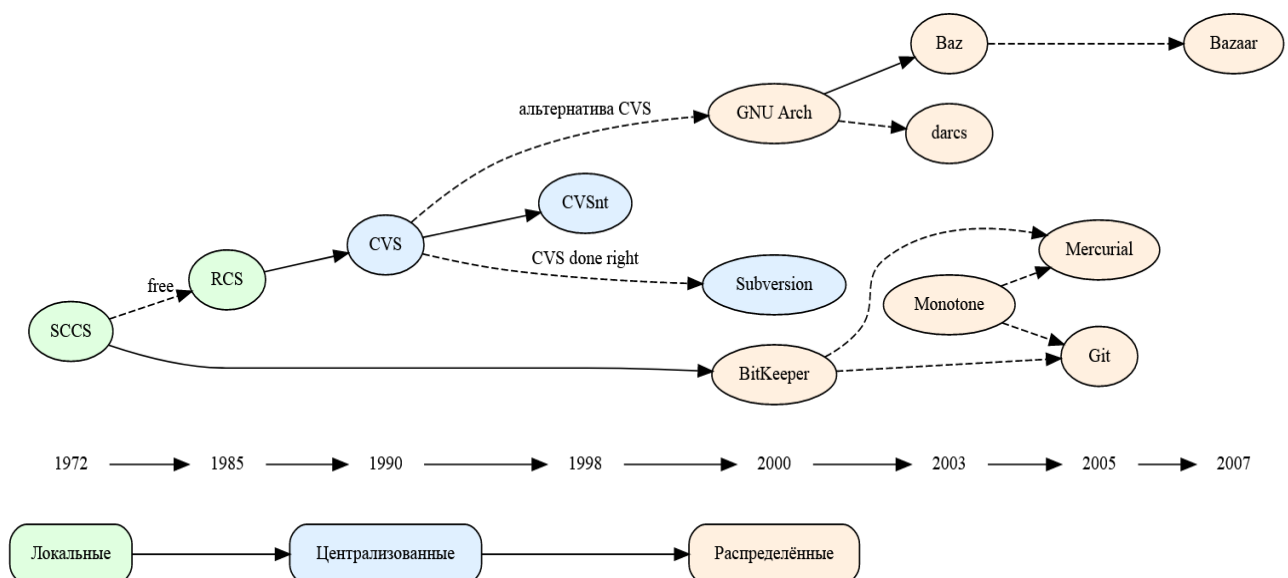
- ветвления в Git;
- создание, слияние веток в Git;
- конфликты при слиянии веток в Git;
- основы работы с удаленными репозиториями;
- совместная работа над проектом (коллаборация).

1. На прошлой лекции

Системы контроля версий

Система управления **версиями** (от англ. *VersionControl System*, *VCS* или *Revision Control System*, *RCS*) – программное обеспечение для облегчения работы с изменяющейся информацией и разработки проекта совместно с коллегами.

История:



Ссылка:

<https://yourcmc.ru/wiki/images/generated/graph/5/54/54d518e63f6f25f3a2a3c90a1ab664d9/graph.source.svg>

Что такое Git

- Git – распределенная система контроля версий;
- поддерживается автономная работа (локальные фиксации изменений могут быть отправлены в репозиторий позже);
- каждое рабочее дерево в Git содержит хранилище с полной историей проекта;
- ни одно хранилище Git не является по своей природе более важным, чем любое другое;
- целостность Git: все объекты сохраняются в базу данных не по имени, а по хеш-сумме содержимого объекта;
- удобный и интуитивно понятный набор команд;
- гибкая система ветвления проектов и слияния веток между собой;
- почти все операции выполняются локально;
- высокая производительность: скорость работы в Git кажется мгновенной;
- универсальный сетевой доступ с использованием протоколов http, ftp, rsync, ssh и др.

Основные определения

Репозиторий Git:	Git хранит информацию в структуре данных, называемой репозиторий (repository). Репозиторий хранится в папке проекта – в папке .git
Репозиторий хранит:	<ul style="list-style-type: none">– набор коммитов (commit objects)– набор ссылок на коммиты (heads)
Commit objects содержат:	<ul style="list-style-type: none">– набор файлов, отображающий состояние проекта в текущий момент времени– ссылки на родительские commit objects– SHA1-имя – 40 символьная строка, которая уникально идентифицирует commit object

Основные команды

```
git init – создание репозитория
git add <имена файлов> – добавляет файлы в индекс
git commit – выполняет коммит проиндексированных файлов в репозиторий
git status – показывает какие файлы изменились между текущей стадией и HEAD. Файлы разделяются на 3 категории: новые файлы, измененные файлы, добавленные новые файлы
git checkout <SHA1 или метка> – получение указанной версии файла
git push – отправка изменений в удаленный репозиторий
git fetch – получение изменений из удаленного репозитория
git clone <remote url> – клонирование удаленного репозитория себе
```

Что такое Github

Github – крупнейший веб-сервис онлайн-хостинга репозиторий, используется для хостинга IT-проектов и их совместной разработки. Основан на системе контроля версий **Git**.

Github

- бесплатный сервис для проектов с открытым исходным кодом;
- место сотрудничества миллионов разработчиков и проектов (кроме размещения кода участники могут общаться, комментировать правки друг друга, а также следить за новостями знакомых); GitHub – социальная сеть для разработчиков;
- предоставляет удобный интерфейс для совместной разработки проектов и может отображать вклад каждого участника в виде дерева;
- дополнительно для проектов есть личные страницы, вики-разметка и система отслеживания ошибок;
- предоставляет возможность прямо на сайте просмотреть файлы проектов с подсветкой синтаксиса для большинства языков программирования.

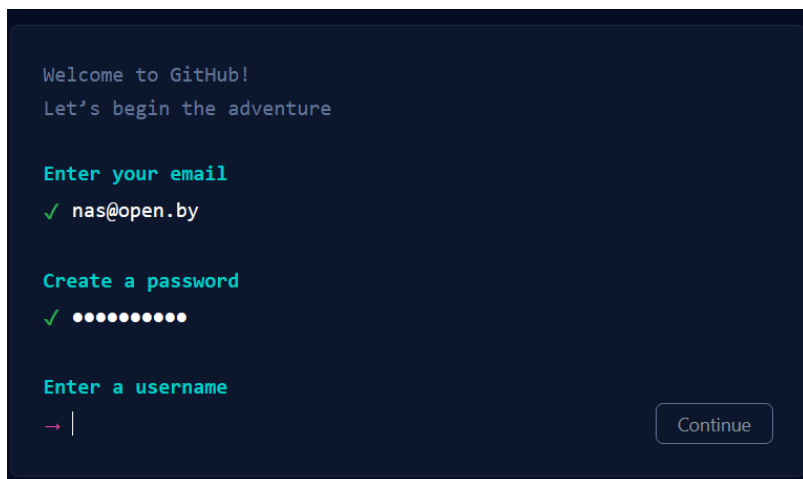
Чем отличается **Git** и **GitHub**

Git:	– инструмент, позволяющий реализовать распределённую систему контроля версий.
GitHub:	– сервис для проектов, использующих Git.

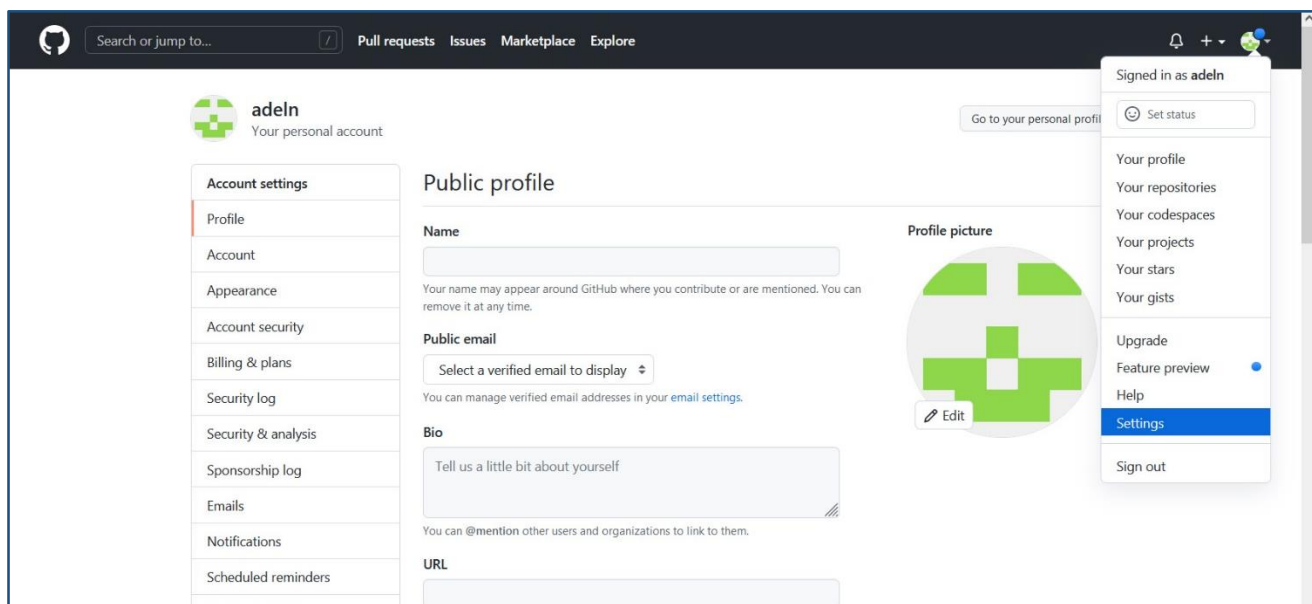
2. GitHub – Сопровождение проекта

Настройка и конфигурация учетной записи

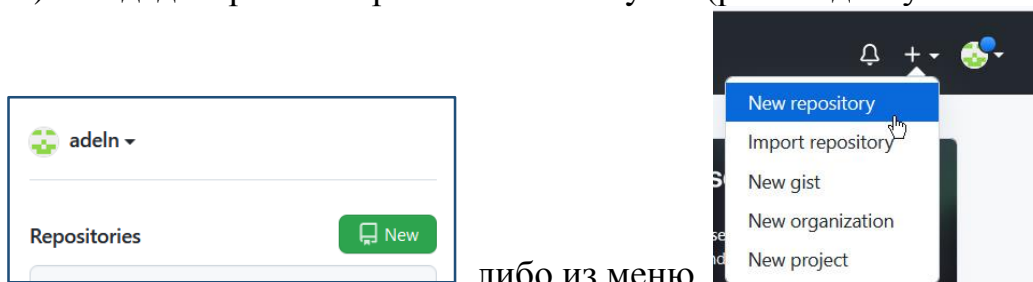
- создать бесплатную учётную запись на сайте <https://github.com/>
- выбрать логин, который еще не занят, указать адрес вашей электронной почты и пароль.



- на следующем шаге (по желанию) можно настроить ваш профиль во вкладке «Settings»

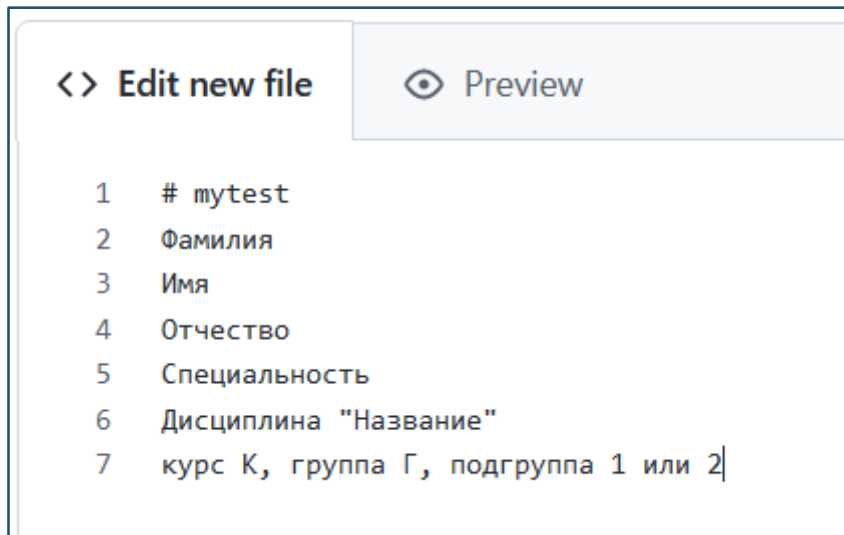
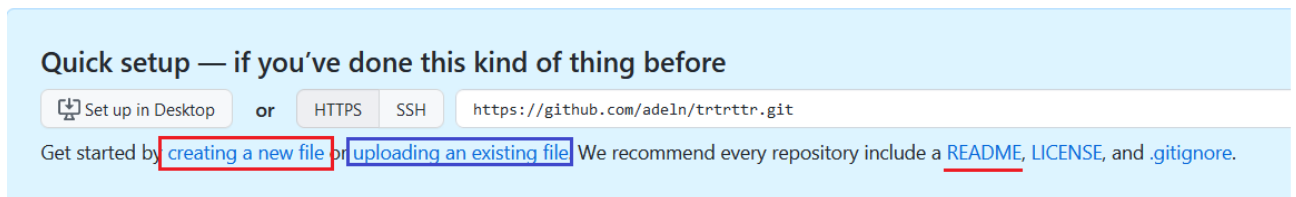


- Создадим репозиторий с именем mytest (режим доступа Public):

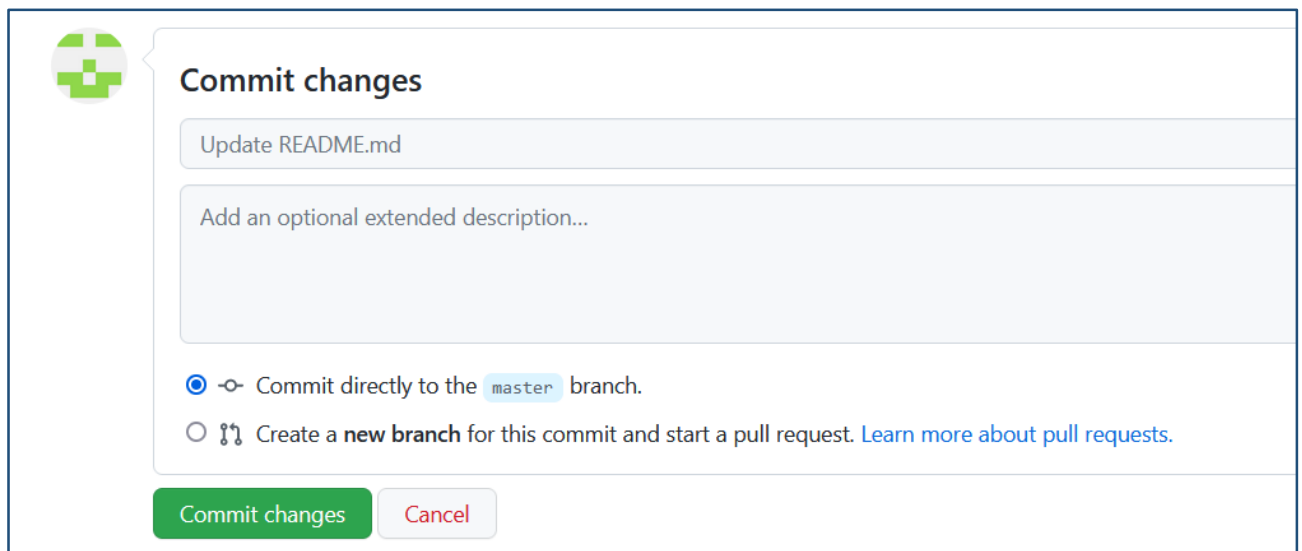


либо из меню



е) Добавим в репозиторий README.md – readme-файл, используя web-интерфейс:




ф) Зафиксируем изменения:





g) Можно отследить изменения файла, выбрав этот файл и кликнув по ссылке:


	Collaboration.docx	Create issue 1	14 hours ago
	README.md	<u>Update README.md</u>	16 hours ago

Update README.md

 master

 **adeln** committed 7 days ago Verified

 Showing **1 changed file** with **1 addition** and **0 deletions**.

✓ ↕ 1 ■■■■ README.md 

↑	@@ -2,6 +2,7 @@
2	2 Фамилия
3	3 Имя
4	4 Отчество
5	+ Специальность
5	6 Дисциплина "Название"
6	7 курс К, группа Г, подгруппа 1 или 2
7	8

Клонирование существующего репозитория в Git

Для получения локальной копии существующего Git-репозитория нужно использовать команду `git clone`

Git получает копию практически всех данных, которые есть на сервере.

При выполнении `git clone` с сервера выгружается версия каждого файла из истории проекта.

Т.о., если сервер выйдет из строя, то можно использовать любой из клонов на любом из клиентов, для того, чтобы вернуть сервер в то состояние, в котором он находился в момент клонирования – все данные, помещенные под версионный контроль, будут сохранены

а) Клонирование репозитория в Git осуществляется командой `git clone <url>`:

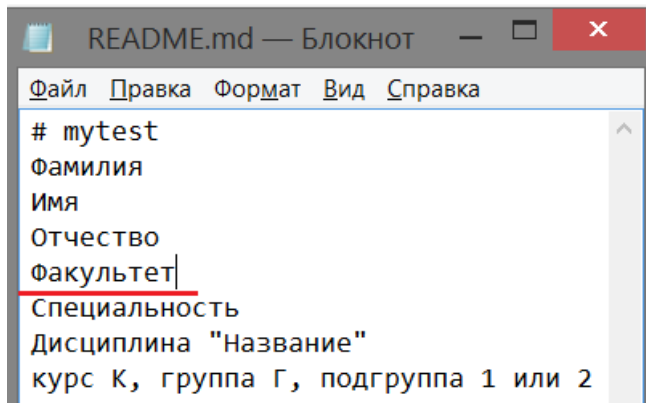
```
$ git clone https://github.com/adeln/mytest
```

```
chimaera@W520-MTNGW32 /d/Ade1/Кафедра/ОПИ+ТРО/Примеры к лабораторным работам/С1
$ git clone https://github.com/adeln/mytest
Cloning into 'mytest'...
remote: Enumerating objects: 39, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 39 (delta 1), reused 0 (delta 0), pack-reused 33
Receiving objects: 100% (39/39), 10.79 KiB | 1.35 MiB/s, done.
Resolving deltas: 100% (6/6), done.
```

Просмотр конфигурационного файла:

```
chimaera@W520-MTNGW32 /d/Ade1/Кафедра/ОПИ+ТРО/Примеры к лабораторным работам/С1
$ git config -l
pack.packsizelimit=2g
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files (x86)/Git/mingw32/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=adel
user.email=narkevich.adelina@gmail.com
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
remote.origin.url=https://github.com/adeln/mytest
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.master.remote=origin
branch.master.merge=refs/heads/master
```

б) Внесем изменения в файл README.md



Просмотр содержимого файла README.md в локальной репозитории с помощью программы cat:

```
one/mytest (master)
$ cat README.md
# mytest
Фамилия
Имя
Отчество
Факультет
Специальность
Дисциплина "Название"
курс К, группа Г, подгруппа 1 или 2
```

На скриншоте просмотр *состояния* репозитория, *индексирование* изменений файла README.md и просмотр *текущего состояния* репозитория:

```
one/mytest (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

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

chimaera@W520 MINGW32 /d/Adel/Кафедра/ОПИ+ТРПО/Примеры к лабораторным работам/С1
one/mytest (master)
$ git add .

chimaera@W520 MINGW32 /d/Adel/Кафедра/ОПИ+ТРПО/Примеры к лабораторным работам/С1
one/mytest (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
```


Фиксация изменений в репозиторий:

```
$ git commit -m "Updated from local repo"
[master 8396e75] Updated from local repo
1 file changed, 2 insertions(+), 1 deletion(-)
```

Просмотр удаленных репозиторий в Git

Просмотреть в Git список *настроенных* удаленных репозиторий можно командой

```
$ git remote.
```

Выполнение: выведены названия доступных удаленных репозиторий, где origin - имя по умолчанию, которое Git дает серверу, с которого производилось клонирование:

```
one/mytest (master)
$ git remote
origin
```

Просмотр log файла*:

```
$ git log
commit 8396e75fbbfb1530f42ab5a7fa8ebd3f908a5ed7 (HEAD -> master)
Author: adel <narkevich.adelina@gmail.com>
Date:   Wed Nov 10 17:51:02 2021 +0300

    Updated from local repo

commit 722fa55b187e36d72eb4dc9fd87b873d153524c5 (origin/master, origin/HEAD)
Author: adel <narkevich.adelina@gmail.com>
Date:   Wed Nov 3 22:31:57 2021 +0300

    Update README.md

commit 61abfc0b0826daf0b6915a89a6ed218a96c8f1dd
Author: adel <narkevich.adelina@gmail.com>
Date:   Wed Nov 3 14:44:51 2021 +0300

    Update README.md

commit 549b29e35675e42c9cddc13b8860d0e653180260
Author: adel <narkevich.adelina@gmail.com>
Date:   Fri Sep 9 18:56:00 2016 +0300
```

Отправка изменений в удаленный репозиторий (Push)

Поделиться своими работами и отправить их в удаленный репозиторий можно с помощью команды **git push**. Формат команды:

```
git push <имя_удаленного_репозитория> <имя_локальной_ветки>.
```

Выполним отправку изменений в удаленный репозиторий:

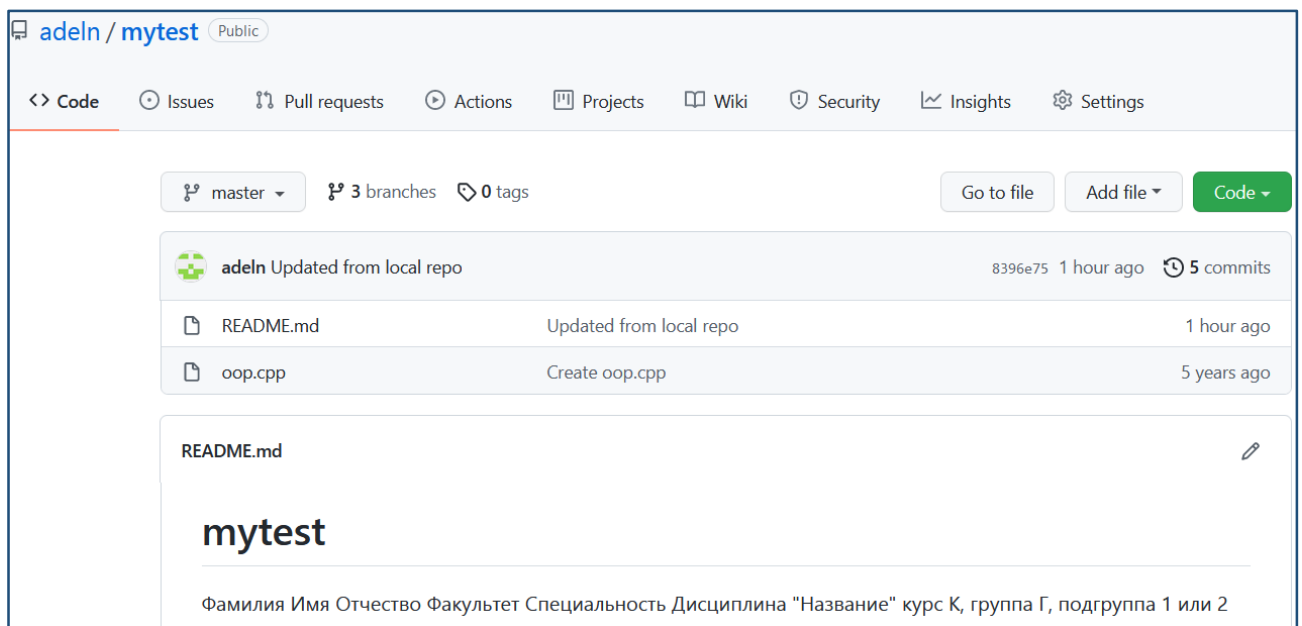
```
$ git push origin master
```

Эта команда выполнится только в случае, если с сервера клонирован репозиторий, к которому у вас есть права на запись (в нашем случае удаленный репозиторий с правами доступа Public)

Результат успешного выполнения команды:

```
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 338 bytes | 338.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/adeln/mytest
722fa55..8396e75 master -> master
```

Присмотр удаленного репозитория:



Указано, что файл `README.md` изменен в локальном репозитории и эти изменения зафиксированы (коммит `8396e75fbbfb1530f42ab5a7fa8ebd3f908a5ed7`, см. лог-файл выше, помеченный *)

Получение изменений из удаленного репозитория — Fetch и Pull

Изменим файл `Readme.md` в удаленном репозитории через web-интерфейс, добавив текущую дату и время.

Зафиксируем изменения и «заберем» изменения из удаленного репозитория в локальный.

формат команды fetch: `git fetch <URL_удаленного_репозитория>`

Выполним команду:

```
$ git fetch https://github.com/adeln/mytest
```

```
$ git fetch https://github.com/adeln/mytest
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 811 bytes | 202.00 KiB/s, done.
From https://github.com/adeln/mytest
* branch                HEAD      -> FETCH_HEAD
```

По команде выполняется обращение к указанному удаленному репозиторию и забираются все те данные проекта, которых у вас ещё нет. При этом их слияния с вашими наработками не происходит и то, над чем вы работаете в данный момент, не модифицируется.

```
$ git show
commit 8396e75fbbfb1530f42ab5a7fa8ebd3f908a5ed7 (HEAD -> master, origin/master, origin/HEAD)
Author: adel <narkevich.adelina@gmail.com>
Date:   Wed Nov 10 17:51:02 2021 +0300

    Updated from local repo

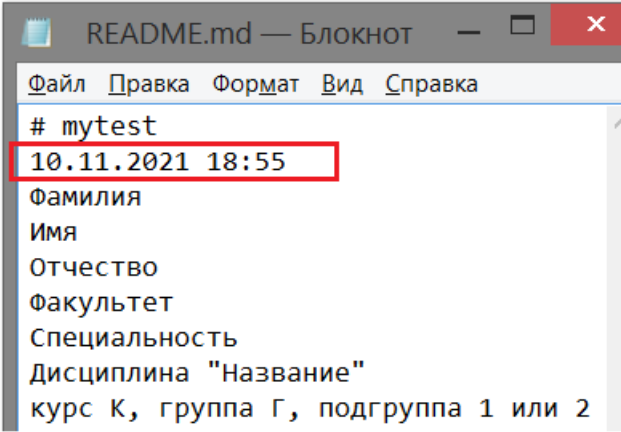
diff --git a/README.md b/README.md
index 2092e47..b61892c 100644
--- a/README.md
+++ b/README.md
@@ -1,7 +1,8 @@
-# mytest
+# mytest
  фамилия
  имя
  Отчество
+Факультет
  Специальность
  Дисциплина "Название"
  курс К, группа Г, подгруппа 1 или 2
```

Можно использовать команду `git pull` чтобы автоматически получить изменения из удалённой ветки и слить их со своей текущей:

```
$ git pull https://github.com/adeln/mytest
From https://github.com/adeln/mytest
* branch          HEAD          -> FETCH_HEAD
Updating 8396e75..b0ec98f
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
```

Изменения в папке локального репозитория:

Имя	Дата изменения	Тип	Размер
.git	10.11.2021 19:23	Папка с файлами	
oop.cpp	10.11.2021 17:30	Файл "CPP"	1 КБ
README.md	<u>10.11.2021 19:23</u>	Файл "MD"	1 КБ



Просмотр лог-файла:

```
$ git log
commit b0ec98f9866f630569e93558425ef0d668d1a725 (HEAD -> master)
Author: adeln <narkevich.adelina@gmail.com>
Date:   Wed Nov 10 18:55:16 2021 +0300

    Update README.md

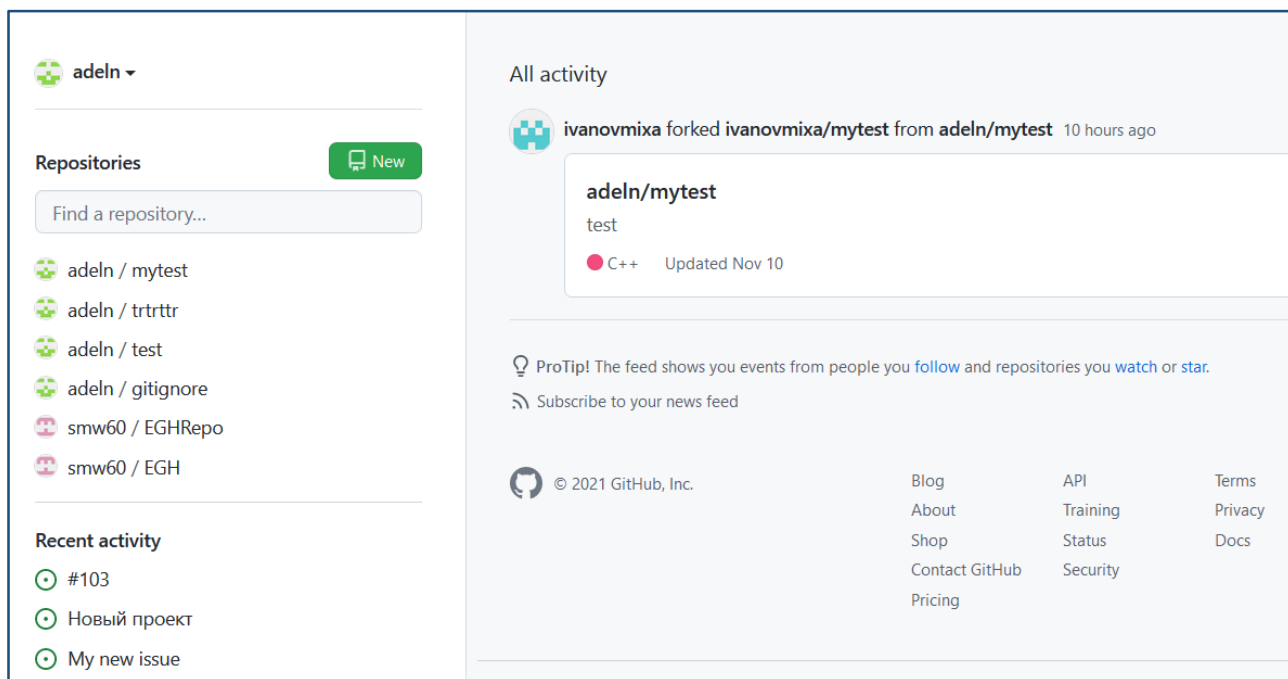
commit 8396e75fbbfb1530f42ab5a7fa8ebd3f908a5ed7 (origin/master, origin/HEAD)
Author: adel <narkevich.adelina@gmail.com>
Date:   Wed Nov 10 17:51:02 2021 +0300

    Updated from local repo
```

3. Совместная работа над проектом

Совместная работа с репозиторием требуется, когда необходимо учитывать текущие задачи, выполнять требования к ним и исправлять баги.

а. На главной странице аккаунта отображается содержимое и все текущие активности:



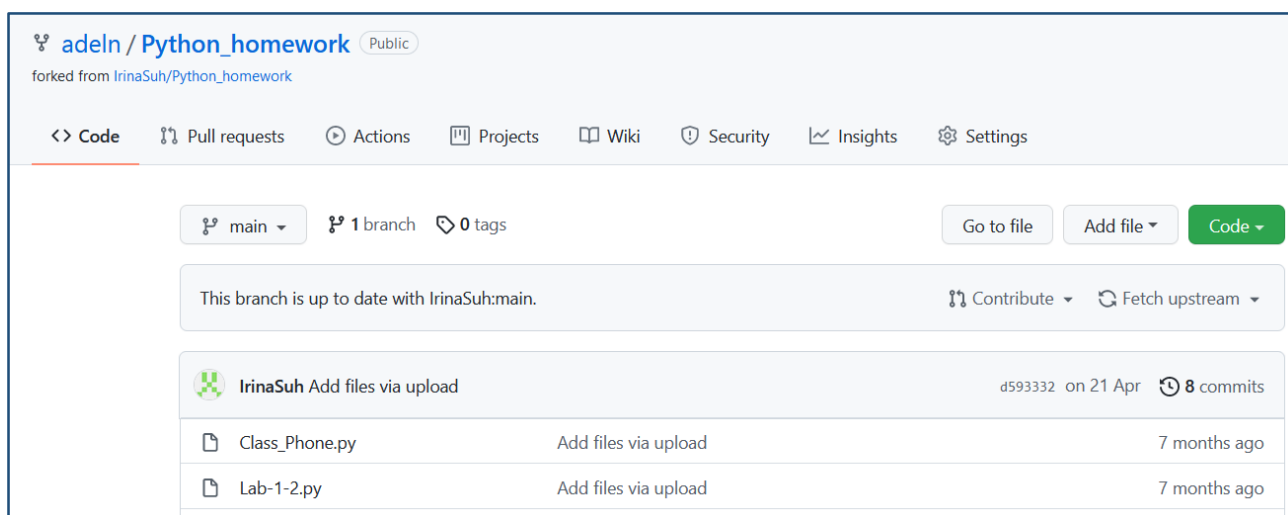
б. Копирование репозитория в Github.

Последовательность действий:

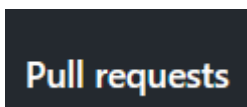
- *создайте форк репозитория коллеги:*

Нажать кнопку  в верхнем правом заголовка проекта коллеги.

Репозиторий коллеги клонирован в отдельную ветку:



- внесите изменения в своей ветке форка (например, добавив какой-нибудь файл).
- создайте запрос (pull-реквест) в репозиторий коллеги, предложив свои изменения:

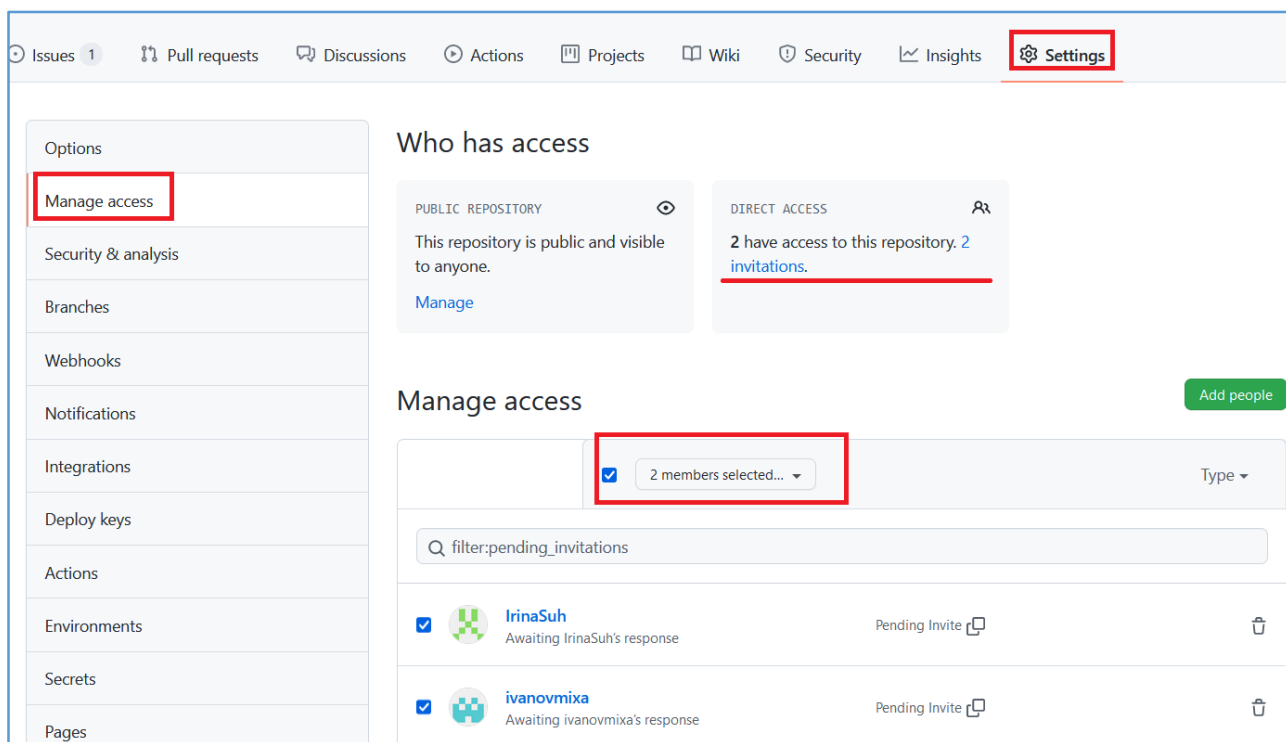


- после слияния ваших изменений в исходный репозиторий его владельцем, заберите в свой форк последние изменения.

с. Добавление членов команды: организация и соавторы

Существует два способа настройки Github для совместной работы:

- **Организации.** Владелец организации может создавать множество команд с разными уровнями доступа для различных репозиториев.
- **Сотрудники.** Владелец репозитория может добавлять коллабораторов с доступом Read + Write для одного репозитория.



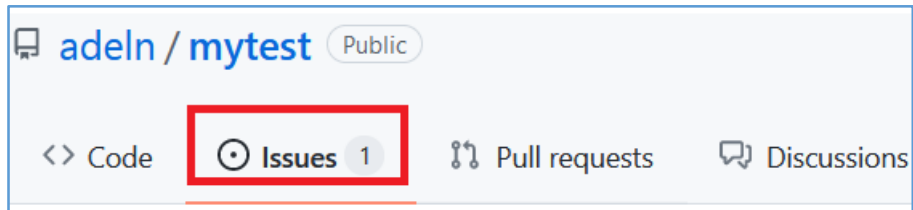
d. Создание проблемы (issue)

Для этого нужно включить вкладку issues.

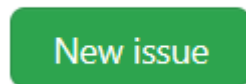
Сделать это можно так: настройка проекта, отметить галочку **issues**. Появляется вкладка **issues**, с помощью которой можно ставить задачи и обсуждать их.

Перейти на страницу репозитория.

Под заголовком репозитория выбрать меню **Issues**:



Нажать кнопку



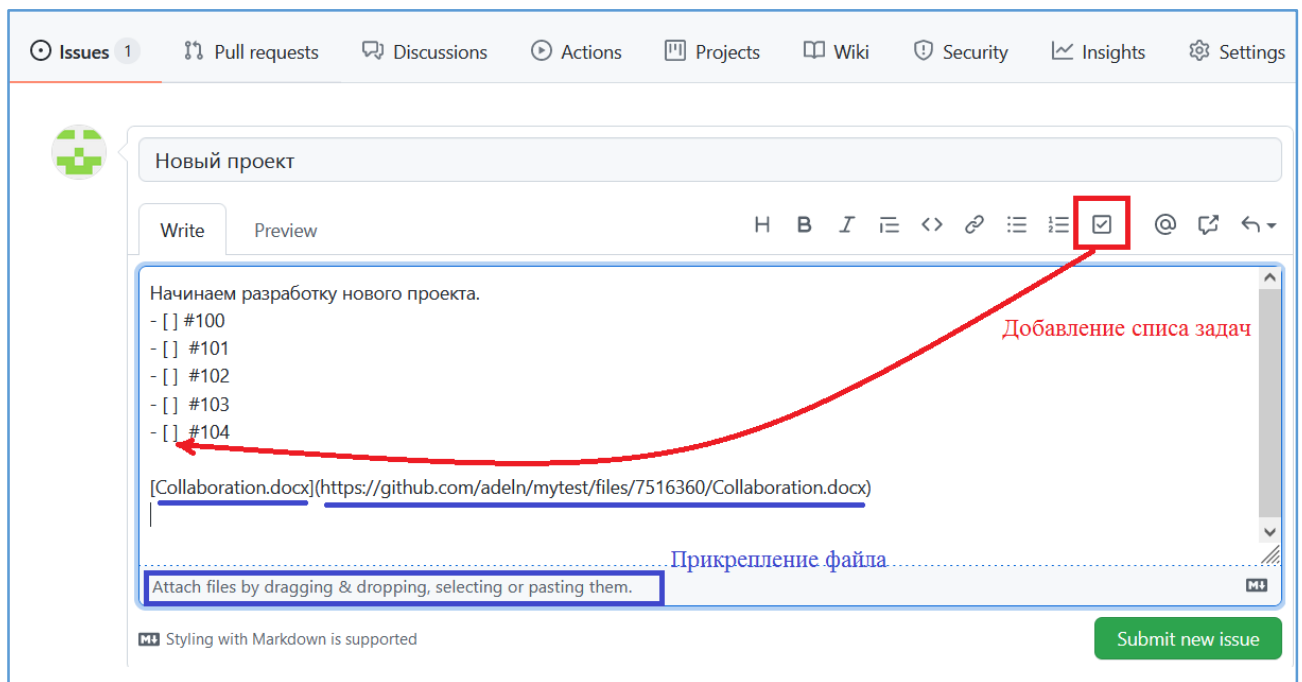
е. Заполнение информации

В поле заголовка **даем** проблеме **название** (название должно отражать суть проблемы).

В закладке Write рабочей области ввода **добавляем** текстовое **описание**, объясняющее цель проблемы, включая любые подробности, которые могут помочь решить проблему. В нашем случае иницилируем начало разработки нового проекта. Определяем цель и ожидаемый результат.

Добавляем список задач – этапы разработки проекта (перед каждым элементом списка надо поставить символ []). Элементы списка могут быть обычным текстом или ссылками на существующие проблемы по их номеру либо по URL. Текст можно отформатировать.

Так же можно прикрепить файл с дополнительной информацией.



Прикрепить файл можно несколькими способами:

- перетащить его из папки;
- выбрать из пункта «выбрать файл»

Имя	Дата изменения	Тип	Размер
.git	10.11.2021 20:41	Папка с файлами	
Collaboration.docx	10.11.2021 20:31	Документ Microso...	13 КБ
README.md	10.11.2021 19:23	Файл "MD"	1 КБ

Issues 1

Pull requests

Actions

Projects

Wiki

Security


Insights

Settings

My new issue #1

Open

adeln opened this issue 2 minutes ago · 0 comments



adeln commented 2 minutes ago

// Заголовок

Owner


...

issue 1

// Описание задачи (Body)










Давайте назовем проект "Новый проект". Как вам? Есть другие предложения?

Описание в файле



Write

Preview

H B I         

Этап 1

// Прикрепленный файл

[Collaboration.docx](https://github.com/adeln/mytest/files/7515148/Collaboration.docx)

Attach files by dragging & dropping, selecting or pasting them.

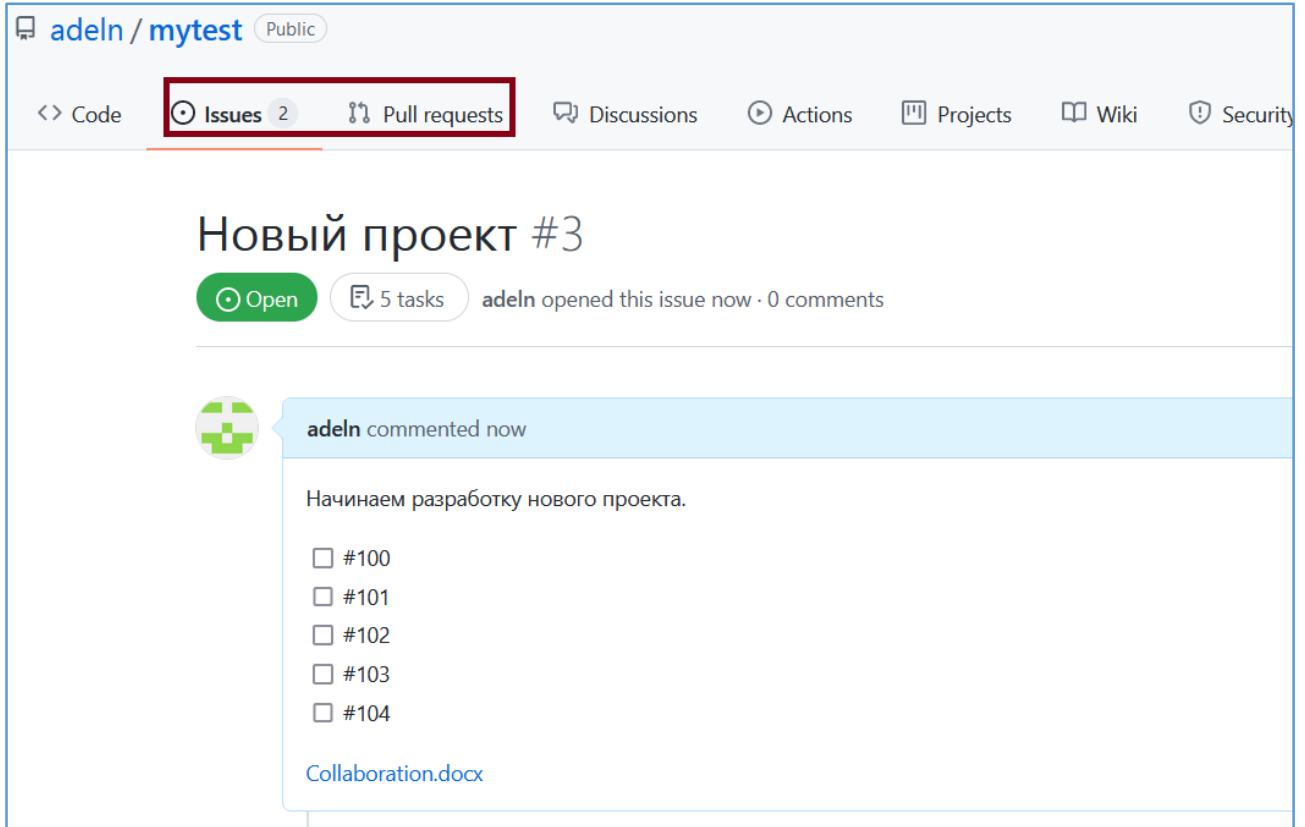
Close with comment

Comment

ф. Назначение проблем и задач другим пользователям

Можно назначить до 10 человек для решения каждой проблемы, включая вас самих, и всех, у кого есть разрешения на запись в репозиторий.

Создаем новый проект, добавляем его описание и список задач, каждой из которых можно назначить исполнителя.



Отображение назначенной проблемы у соавтора:

