

Мануал по использованию Git и GitHub в проекте

Оглавление

Инструкция по авторизации	2
GitHub CLI.....	2
SSH	4
Инструкция по пользованию (командная строка).....	6
Предварительная настройка	6
Начало работы	6
Внесение изменений	7
Скачивание изменений	8
Инструкция по пользованию (GitHub CLI).....	9
Предварительная настройка	9
Начало работы	10
Внесение изменений	11
Скачивание изменений	12
Важные вопросы по оформлению	13
Как называть новые ветки?	13
Какого стиля придерживаться в комментариях к коммитам?	13
Как оформлять заголовок коммита?	13
Как оформлять развернутое описание (тело) коммита?	13

Инструкция по авторизации

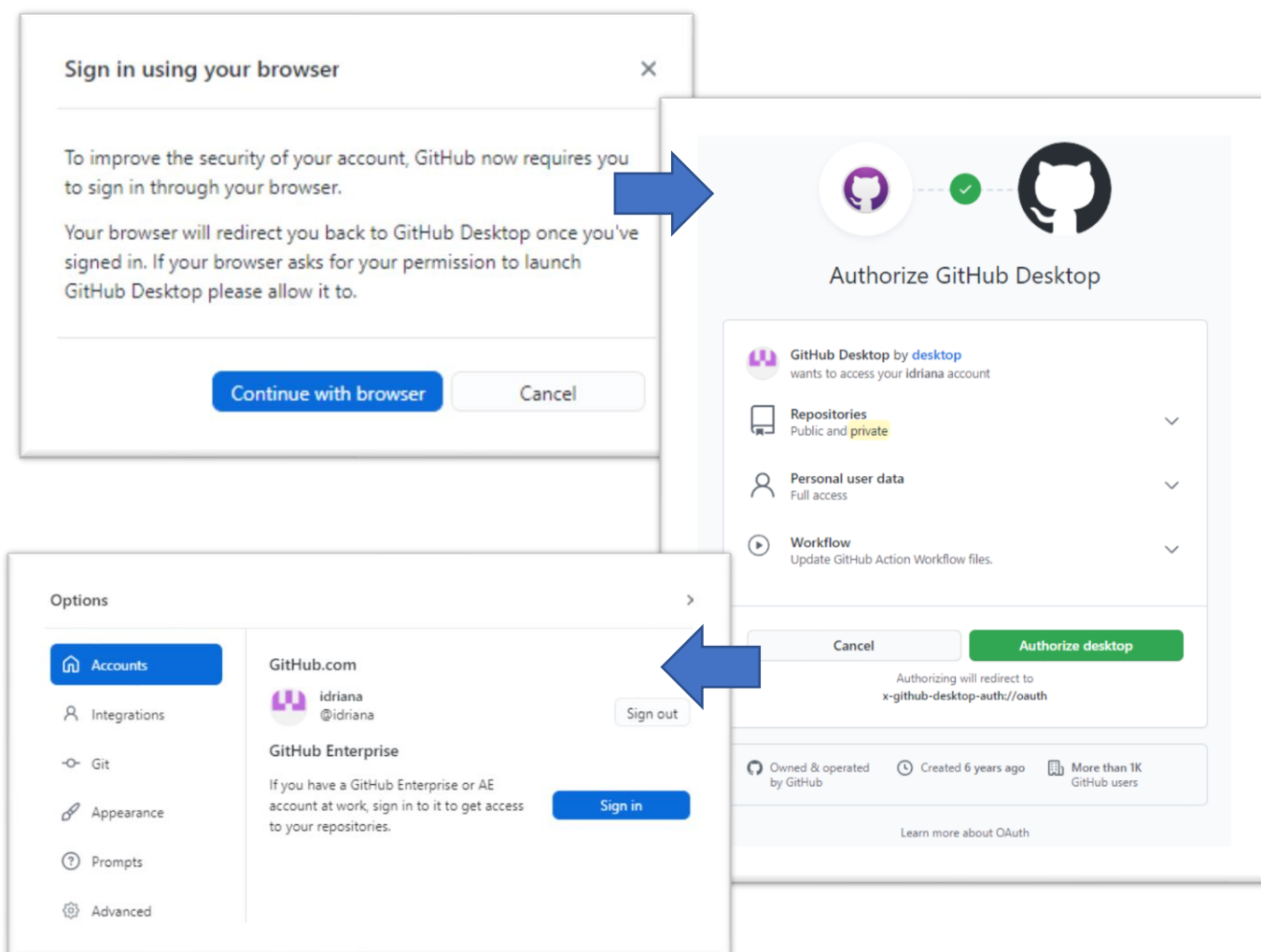
Для начала работы каждому участнику понадобятся следующее:

- [Unity](#)
- [Git](#)
- Аккаунт на [GitHub](#)
- Быть добавленным в репозиторий проекта

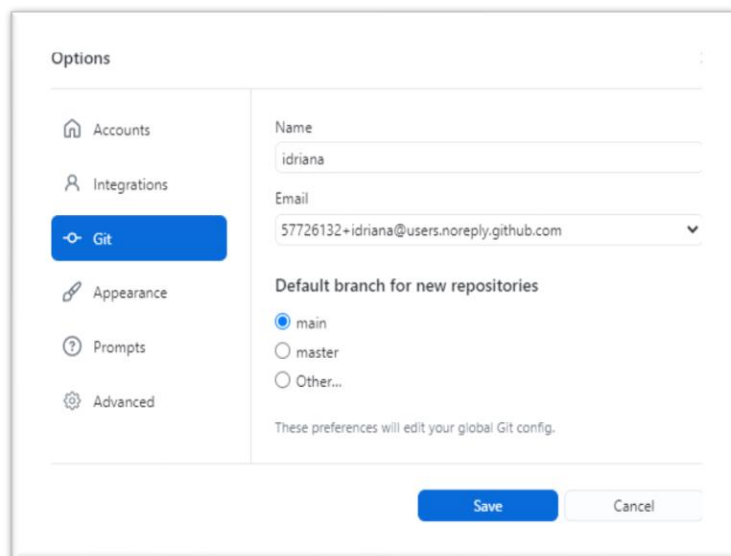
GitHub CLI

Официальное приложение от GitHub для управления Git, скачать можно [здесь](#).

Первым делом необходимо запустить приложение и авторизоваться через GitHub. Приложение предложит открыть браузер и разрешить доступ к аккаунту, после чего уже браузер предложит перейти в приложение. Управлять аккаунтами можно в настройках приложения:



Приложение автоматически настроит Git, введя информацию о пользователе, разрешая пользователю действовать от имени своей учетной записи:



Options

Accounts

Integrations

Git

Appearance

Prompts

Advanced

Name
idriana

Email
57726132+idriana@users.noreply.github.com

Default branch for new repositories

☒ main

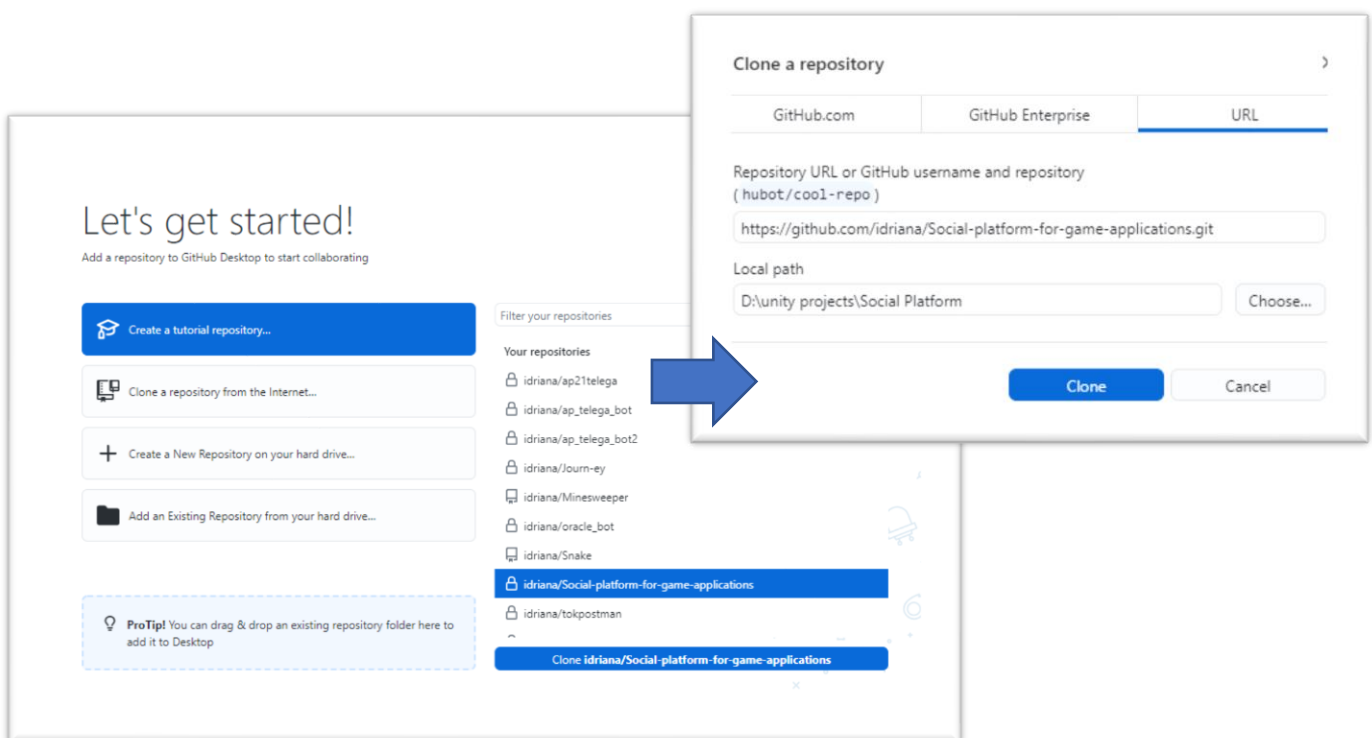
☐ master

☐ Other...

These preferences will edit your global Git config.

Save Cancel

Далее необходимо найти проект и клонировать его, указав в открывшемся окне путь до папки с проектами Unity:



Let's get started!

Add a repository to GitHub Desktop to start collaborating

Create a tutorial repository...

Clone a repository from the Internet...

Create a New Repository on your hard drive...

Add an Existing Repository from your hard drive...

ProTip! You can drag & drop an existing repository folder here to add it to Desktop

Filter your repositories

Your repositories

- idriana/ap21telega
- idriana/ap_teleaga_bot
- idriana/ap_teleaga_bot2
- idriana/Journ-ey
- idriana/Minesweeper
- idriana/oracle_bot
- idriana/Snake
- idriana/Social-platform-for-game-applications**
- idriana/tokpostman

Clone idriana/Social-platform-for-game-applications

Clone a repository

GitHub.com GitHub Enterprise **URL**

Repository URL or GitHub username and repository (hubot/cool-repo)

https://github.com/idriana/Social-platform-for-game-applications.git

Local path

D:\unity projects\Social Platform Choose...

Clone Cancel

Репозиторий импортирован, осталось открыть папку через Unity!

SSH

SSH – это сетевой протокол, позволяющий соединяться с удалённым сервером и выполнять на нём команды. Большинство современных систем поддерживают SSH по умолчанию.

GitHub позволяет привязать к аккаунту SSH ключ для доступа к репозиторию.

Подробнее про это можно прочитать [здесь](#).

Первым делом необходимо создать публичный (для загрузки в GitHub) и приватный (для личного пользования) ключи, я буду использовать RSA. Выполняю команду

ssh-keygen -t rsa -b 4096 -C example@edu.hse.ru

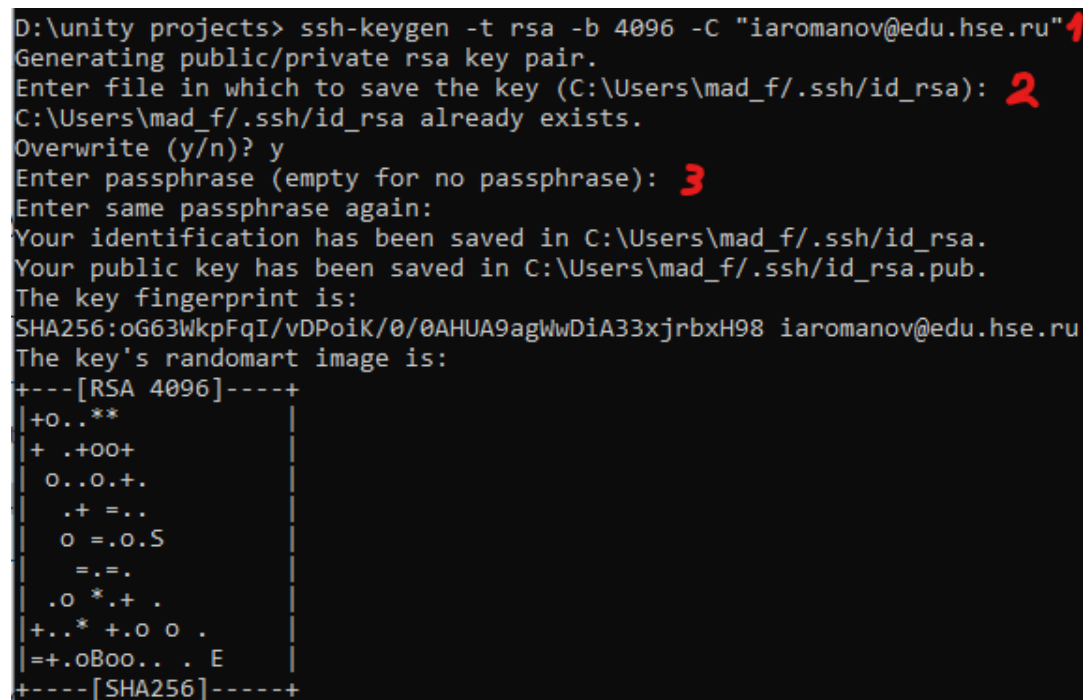
Где

ssh-keygen – команда для генерации пары ключей

-t rsa – флаг и значение, указывающие на алгоритм RSA

-b 4096 – длина ключа

-C example@edu.hse.ru – отметка автора ключа **ПОЧТУ СЛЕДУЕТ ВВЕСТИ СВОЮ**



```
D:\unity projects> ssh-keygen -t rsa -b 4096 -C "iaromanov@edu.hse.ru"
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\mad_f\.ssh\id_rsa): 
C:\Users\mad_f\.ssh\id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase): 
Enter same passphrase again: 
Your identification has been saved in C:\Users\mad_f\.ssh\id_rsa.
Your public key has been saved in C:\Users\mad_f\.ssh\id_rsa.pub.
The key fingerprint is:
SHA256:oG63WkpFqI/vDPoIK/0/0AHUA9agWwDiA33xjrbxH98 iaromanov@edu.hse.ru
The key's randomart image is:
+---[RSA 4096]-----+
|+O..**               |
|+ .+00+              |
| o..O.+              |
| .+ =..              |
| o =.O.S              |
| =.=.                |
| .O *.+ .            |
|+..* +.O O .          |
|+=.oBoo.. . E        |
+---[SHA256]-----+
```

1 – Ввод команды

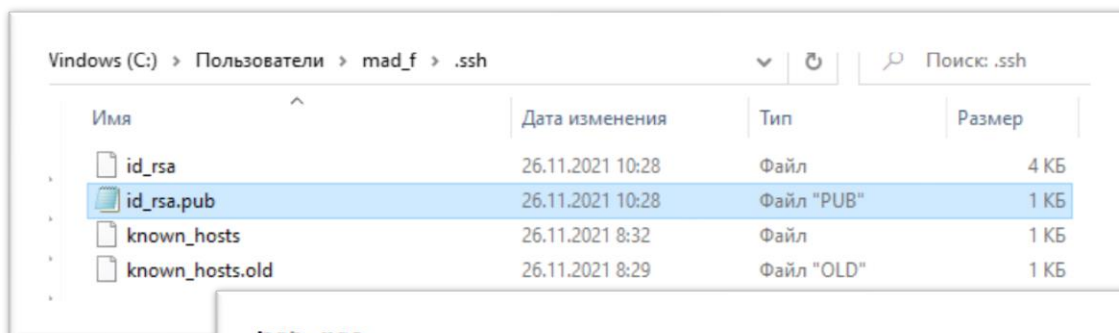
2 – Программа просит ввести файл, в который она запишет ключ. Как правило он указан по умолчанию в настройках системы: «C:/Users/<username>/.ssh/id_rsa» или «~/.ssh/id_rsa». При этом в имени файла после «id_» указывается алгоритм шифрования.

3 – Программа просит ввести пароль для SSH. Стоит ли его устанавливать? Возможно.

Далее идет информация о работе программы.

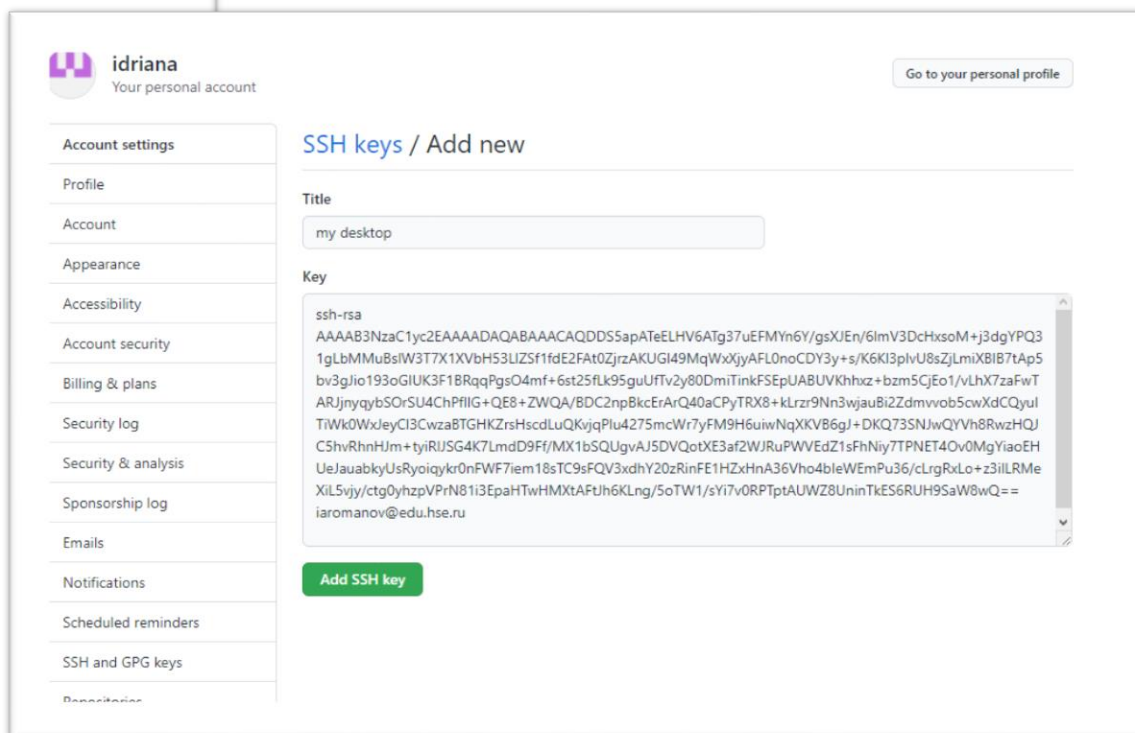
После создания ключей их можно найти в соответствующем каталоге «C:/Users/<username>/ssh» или «~.ssh». Файлы «id_rsa» и «id_rsa.pub» являются публичным и приватным ключом соответственно, для дальнейшей работы нам необходим последний.

Содержимое файла необходимо скопировать **ПОЛНОСТЬЮ**, после чего зайти в настройки аккаунта на сайте GitHub в раздел SSH and GPG keys и добавить новый ключ. Название ключа ни на что не влияет и нужно для того, чтобы пользователь понимал, для чего этот ключ, а сам ключ должен полностью совпадать с содержимым файла «id_rsa.pub»



ssh-rsa

```
AAAAB3NzaC1yc2EAAAADAQABAAQDD55apATeELHV6ATg37uEFMYn6Y/gsXJEn/6lmV3DcHxsoM+j3dgYPQ31g1bMMuBsIW3T7X1XVbH53LlZsf1fdE2FAt0ZjrzAKUGl49MqWxXjyAFL0noCDY3y+s/K6Kl3plvU8sZjLmIXBlB7tAp5bv3gJio193oGIUK3F1BRqgPgsO4mf+6st25fLk95guUftv2y80DmiTinkFSEpUABUVKhxz+bzm5CjEo1/vLhX7zaFwTARJjnyqybSOrSU4ChPfllG+QE8+ZWQA/BDC2npBkcErArQ40aCPyTRX8+kLrZr9Nn3wjauBi2Zdmvob5cwXdcQyuITiWk0WxJeyCl3CwzaBTGHKZrsHscdLuQKvjPlu4275mcWr7yFM9H6uiwNqXKVB6gJ+DKQ73SNJwQYVh8RwzHQJC5hvRhHJm+tyiRIJSG4K7LmdD9ff/MX1bSQUgVAJ5DVQotXE3af2WJRuPWVEdZ1sFhNiy7TPNET4Ov0MgYiaoEHUeJauabkyUsRyoiqykr0nFWF7iem18sTC9sFQV3xdhY20zRinFE1HZxHnA36Vho4bIeWEmPu36/cLrgRXL0+z3illRMeXiL5vjy/ctg0YhZpVPrN81i3EpaHTwHMXtAftJh6KLng/5oTW1/sYi7v0RPTptAUWZ8UninTkES6RUH9SaW8wQ== iaromanov@edu.hse.ru
```



Наконец, после сохранения ключа можно скопировать репозиторий с помощью команды (путь указан в способах клонирования репозитория на GitHub)

```
git clone git@github.com:idriana/Social-platform-for-game-applications.git "Social Platform"
```

Где

- git clone – команда для клонирования репозитория

- [git@github.com:idriana/Social-platform-for-game-applications.git](https://github.com/idriana/Social-platform-for-game-applications.git) - путь до репозитория

- "Social Platform" – путь до папки, в которую будут скопированы файлы репозитория; в данном случае команда вызывалась из папки проекта, потому путь представляет собой только название папки

Репозиторий импортирован, осталось открыть папку через Unity!

Инструкция по пользованию (командная строка)

Для наглядности используется Git Bash.

Предварительная настройка

В самом начале необходимо указать Git, кто вносит изменения в проект с помощью двух команд:

```
git config --global user.email "example@edu.hse.ru"
git config --global user.name "Ivan Ivanov"
```

Они служат для указания почты и инициалов соответственно, нужно просто вставить **СВОИ ДАННЫЕ** в кавычки.

```
mad_f@DESKTOP-LR10L7U MINGW64 /d/unity projects/Social Platform (master)
$ git config --global user.email "iaromanov@edu.hse.ru"

mad_f@DESKTOP-LR10L7U MINGW64 /d/unity projects/Social Platform (master)
$ git config --global user.name "Ivan Romanov"
```

Начало работы

Перед тем как начинать работать над новой задачей следует создать новую ветку (команды вводятся из соответствующей директории):

```
git branch CoolBranch master
```

Где

- git branch – команда для создания новой ветки

- CoolBranch – Название новой ветки

- master – Ветка (или коммит), содержимое которой будет скопировано в новую ветку

```
git checkout CoolBranch
```

Где

- git checkout – команда для переключения на другую ветку

- CoolBranch – название ветки, на которую необходимо переключиться

```
mad_f@DESKTOP-LR10L7U MINGW64 /d/unity projects/Social Platform (master)
$ git branch CoolBranch master

mad_f@DESKTOP-LR10L7U MINGW64 /d/unity projects/Social Platform (master)
$ git checkout CoolBranch
Switched to branch 'CoolBranch'

mad_f@DESKTOP-LR10L7U MINGW64 /d/unity projects/Social Platform (CoolBranch)
```

Для каждой новой ветки необходимо указать, к какому удаленному репозиторию она относится, для этого можно воспользоваться командой

git push --set-upstream origin CoolBranch

Где

- git push --set-upstream – комбинация из команды и флага для указания удаленного репозитория
- origin – название удаленного репозитория по умолчанию. Удаленный репозиторий был автоматически получен git при клонировании репозитория.
- CoolBranch – Название новой ветки, для которой указывается удаленный репозиторий

Внесение изменений

Предположим, у нас есть измененный или созданный файл «new file.txt» (статус репозитория всегда можно посмотреть с помощью команды git status). Все что нужно сделать это добавить изменения в git следующей командой:

Git add "new file.txt"

Где

- git add – команда для добавления изменения в git
- "new file.txt" – название файла, который необходимо добавить. Может быть заменено на название директории или точку (git add .) для добавления абсолютно всех новых элементов

```
mad_f@DESKTOP-LR10L7U MINGW64 /d/unity projects/Social Platform (CoolBranch)
$ git status
On branch CoolBranch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    new file.txt

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

mad_f@DESKTOP-LR10L7U MINGW64 /d/unity projects/Social Platform (CoolBranch)
$ git add "new file.txt"

mad_f@DESKTOP-LR10L7U MINGW64 /d/unity projects/Social Platform (CoolBranch)
$ git status
On branch CoolBranch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   new file.txt
```

После добавления изменения его нужно сохранить соответствующей командой:

Git commit

При этом откроется текстовый редактор, в котором в первой строчке следует ввести заголовок коммита, а после двух переходов строки развернутое описание коммита.

```
mad_f@DESKTOP-LR10L7U MINGW64 /d/unity projects/Social Platform (CoolBranch)
$ git commit
[CoolBranch 5e6c7f2] brief description
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 new file.txt
```

```
brief description

full description...
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch CoolBranch
# Changes to be committed:
#   new file:   new file.txt
#
```

Наконец после сохранения изменений их необходимо передать на сервер GitHub командой

Git push

Скачивание изменений

Git позволяет скачивать изменения в удаленном репозитории с помощью команды

Git pull

В лучшем случае git сам обновит файлы проекта и выдаст информацию о методе, который он использовал

```
mad_f@DESKTOP-LR10L7U MINGW64 /d/unity projects/Social-platform-for-game-applica
tions (master)
$ git pull
Updating 1c9d573..209e165
Fast-forward
```

Однако иногда возникают конфликты. В этой ситуации необходимо вручную отредактировать файл и сохранить его, тогда конфликт разрешится и можно будет продолжить обновление командой

Git add <название файла>

```
mad_f@DESKTOP-LR10L7U MINGW64 /d/unity projects/Social platform (master)
$ git pull
Auto-merging new file.txt
CONFLICT (content): Merge conflict in new file.txt
Automatic merge failed; fix conflicts and then commit the result.
```

```
mad_f@DESKTOP-LR10L7U MINGW64 /d/unity projects/Social platform (master|MERGING)
$ git add "new file.txt"
```


Сохраняются изменения как обычно командой:

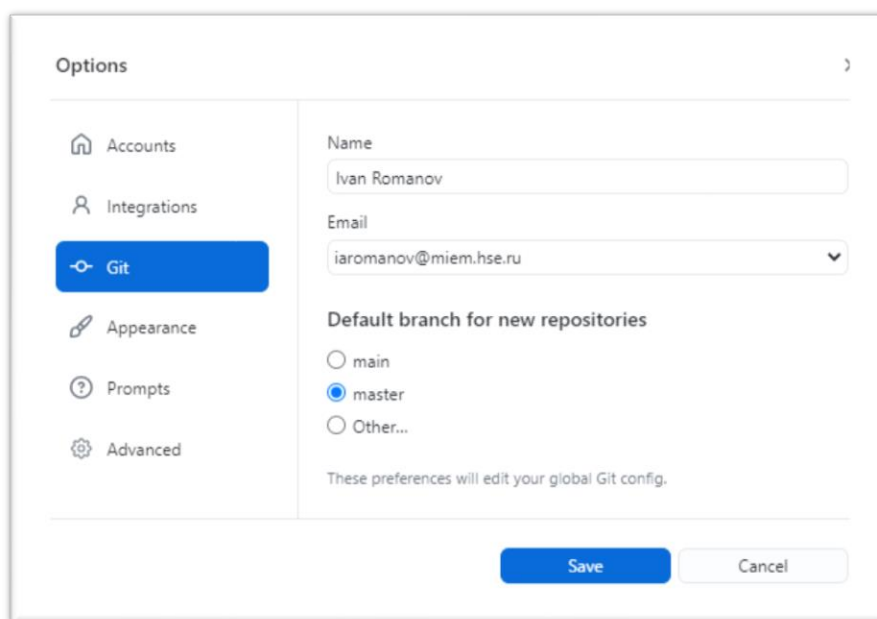
Git commit

```
mad_f@DESKTOP-LR10L7U MINGW64 /d/unity projects/Social platform (master|MERGING)
$ git commit
[master 172ccae] Merge branch 'master' of github.com:idriana/Social-platform-for-game-applications
```

Инструкция по пользованию (GitHub CLI)

Предварительная настройка

Все необходимые данные приложение получит из вашего аккаунта на GitHub, однако настоятельно рекомендуется дополнительно проверить и при необходимости исправить настройки:



Options

- Accounts
- Integrations
- Git**
- Appearance
- Prompts
- Advanced

Name
Ivan Romanov

Email
iaromanov@miem.hse.ru

Default branch for new repositories

☐ main

☒ master

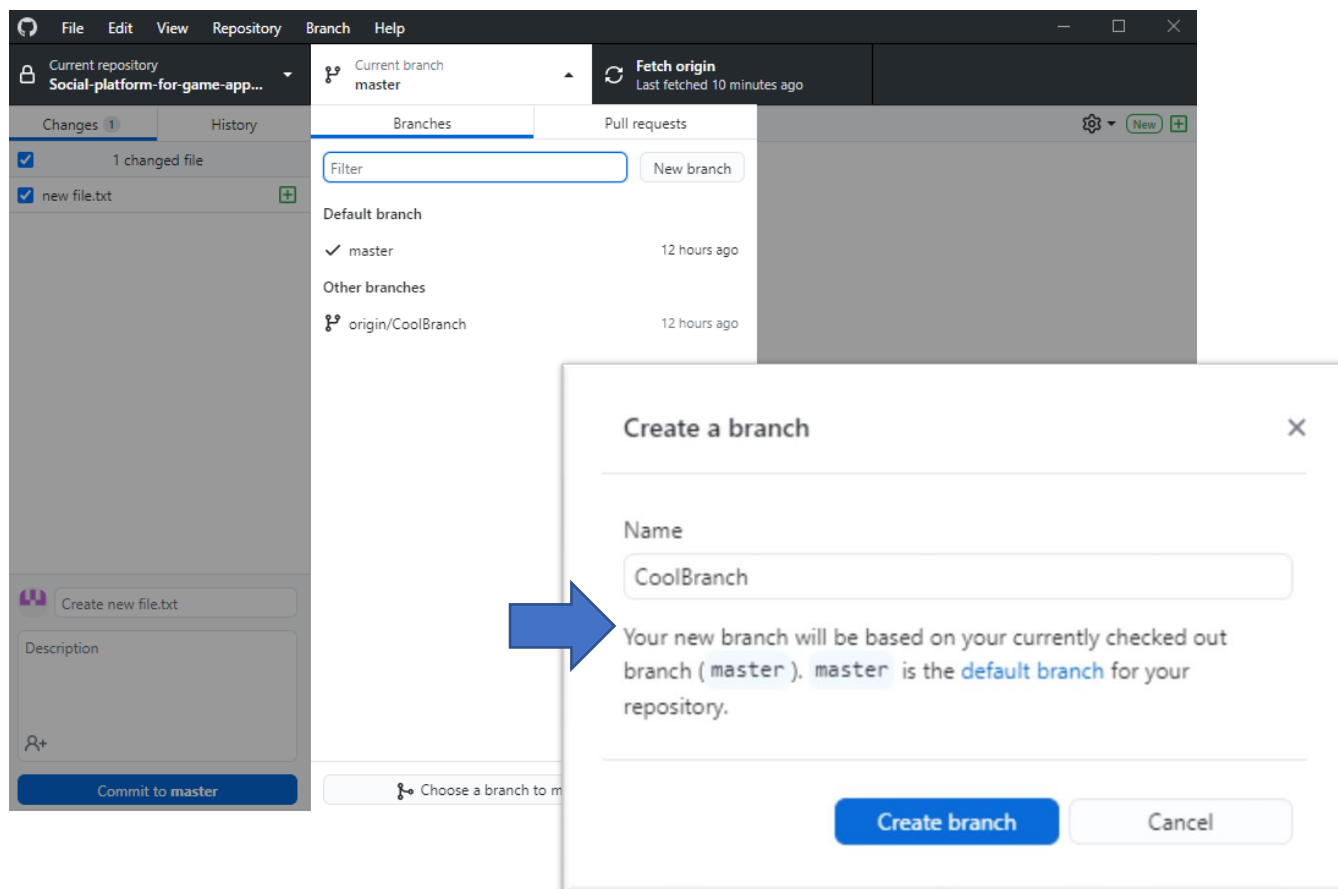
☐ Other...

These preferences will edit your global Git config.

Save Cancel

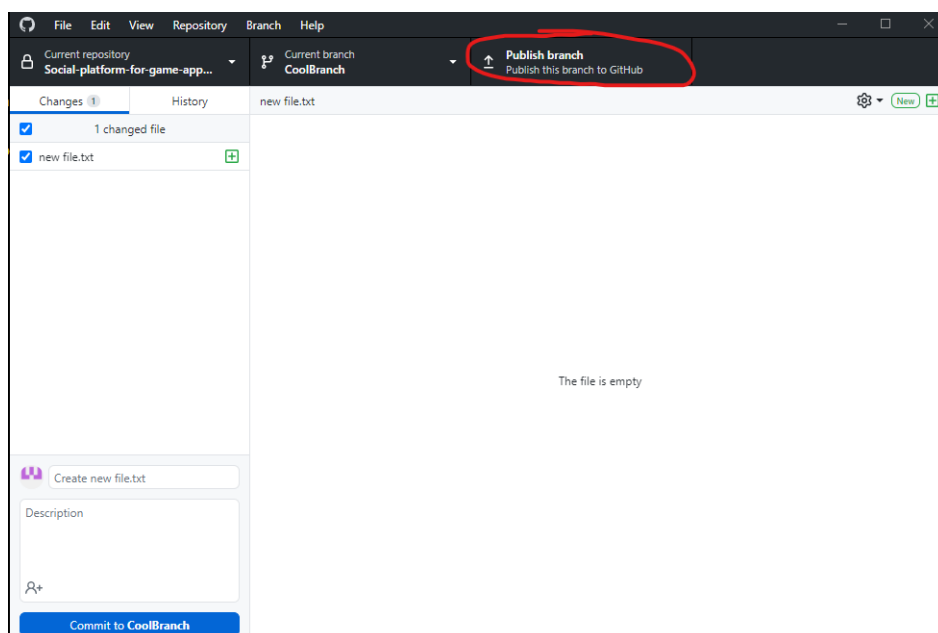
Начало работы

Перед тем как начинать работать над новой задачей следует создать новую ветку:



Содержимое новой ветки будет скопировано из текущей.

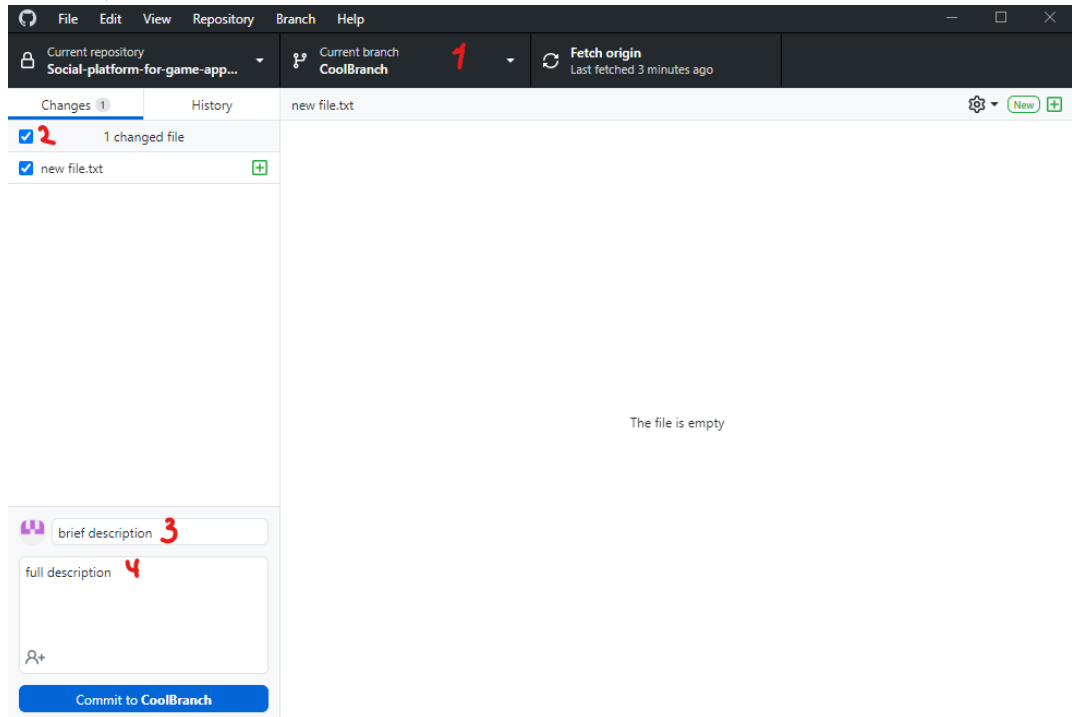
После этого необходимо связать ветку с удаленным репозиторием, нажав на кнопку «publish branch»:



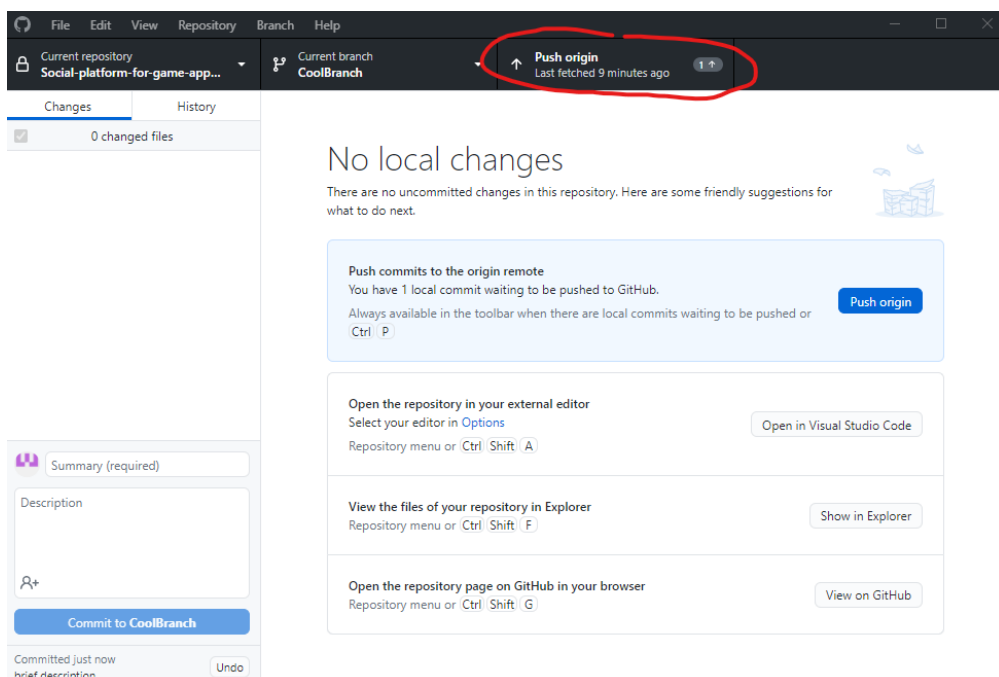
Внесение изменений

Предположим, у нас есть измененный или созданный файл «new file.txt», тогда чтобы сохранить изменения в git необходимо:

1. Проверить, в какую ветку будут сохраняться изменения
2. Выбрать файлы для отправки
3. Ввести заголовок коммита
4. Ввести развернутое описание коммита (опционально)
5. нажать на кнопку «Commit to <название ветки>»

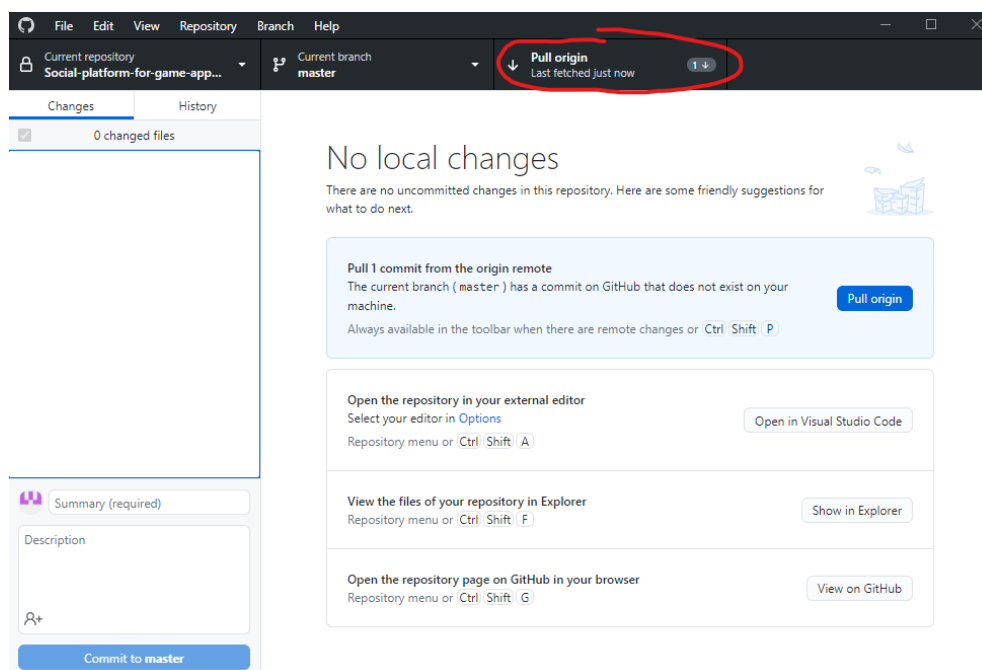


После сохранения изменений их необходимо передать на сервер, нажав на кнопку «Push origin»:

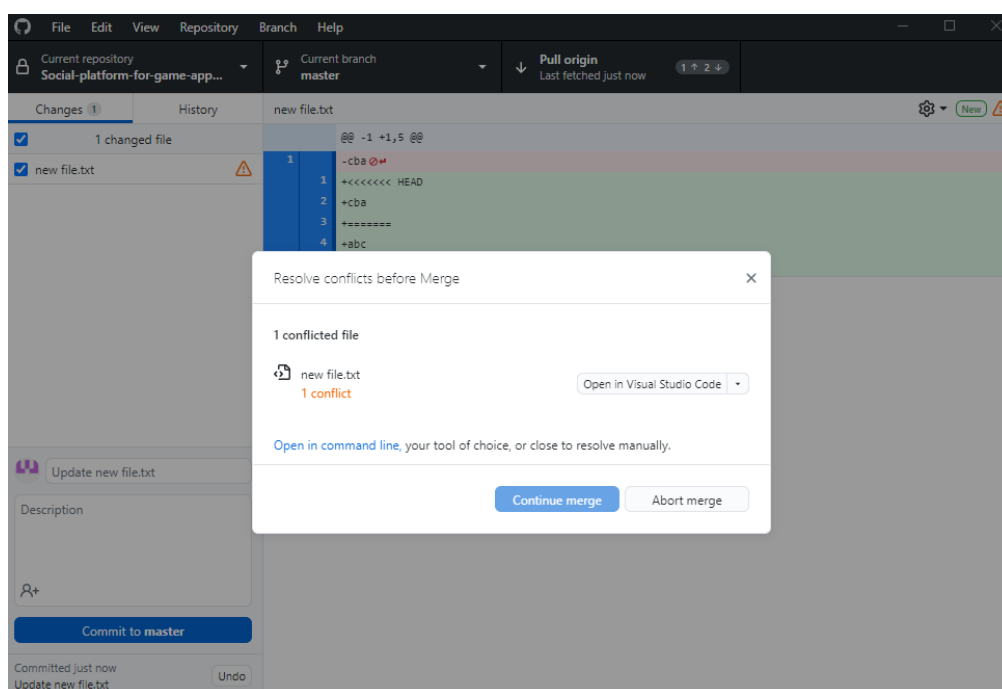


Скачивание изменений

Приложение автоматически проверяет изменения в удаленном репозитории и при возможности предлагает пользователю автоматически скачать их, нажав на кнопку «Pull origin»



В случае пересечения изменений в репозитории и удаленном репозитории, приложение предложит выбрать лучший вариант. Файл будет необходимо отредактировать вручную и сохранить, тогда конфликт разрешится:



Важные вопросы по оформлению

Как называть новые ветки?

Информативно. По названию сразу должно быть понятно, что именно разрабатывается в данной ветке. Если разработка относится только к одному из трёх микросервисов, это тоже следует указать (Auth/Roster/Chat). Например:

- Database – общая ветка для базы данных
- (Auth)API – ветка для API сервиса авторизации
- (<микросервис>)<название ветки> - в общем случае

Какого стиля придерживаться в комментариях к коммитам?

Есть ряд общепринятых норм, которых придерживается сообщество GitHub, подробнее про каждый можно прочитать в статье на хабре:

1. Отделяйте заголовок от тела пустой строкой
2. Ограничивайте заголовок 50 символами
3. Пишите заголовок с заглавной буквы
4. Не ставьте точку в конце заголовка
5. Используйте повелительное наклонение в заголовке
6. Переходите на следующую строку в теле на 72 символах
7. В теле отвечайте на вопросы *что* и *почему*, а не *как*

Как оформлять заголовок коммита?

Комментарии тоже должны быть информативными. По коммиту должно быть сразу понятно, к чему он относится. Например:

- Database: Add chat-to-messages relationship
- (Auth)API: Add tests for chat requests
- <раздел>: <краткая информация о коммите> - в общем случае

Как оформлять развернутое описание (тело комментария) коммита?

Развернутый комментарий должен содержать следующие вещи:

- Проблему, решаемую этим коммитом
- Контекст (опционально)
- Побочные эффекты от коммита

Объяснять какие алгоритмы были использованы в самом коде НЕ следует, код говорит сам за себя.

Далее следует идеальный пример:

```
commit eb0b56b19017ab5c16c745e6da39c53126924ed6
Author: Pieter Wuille <pieter.wuille@gmail.com>
Date:   Fri Aug 1 22:57:55 2014 +0200
```

Simplify serialize.h's exception handling

Remove the 'state' and 'exceptmask' from serialize.h's stream implementations, as well as related methods.

As exceptmask always included 'failbit', and setstate was always called with bits = failbit, all it did was immediately raise an exception. Get rid of those variables, and replace the setstate with direct exception throwing (which also removes some dead code).

As a result, good() is never reached after a failure (there are only 2 calls, one of which is in tests), and can just be replaced by !eof().

fail(), clear(n) and exceptions() are just never called. Delete them.