Отчёт по лабораторной работе

Дисциплина: операционные системы

Пермякова Елизавета Евгеньевна

Содержание

# Цель работы

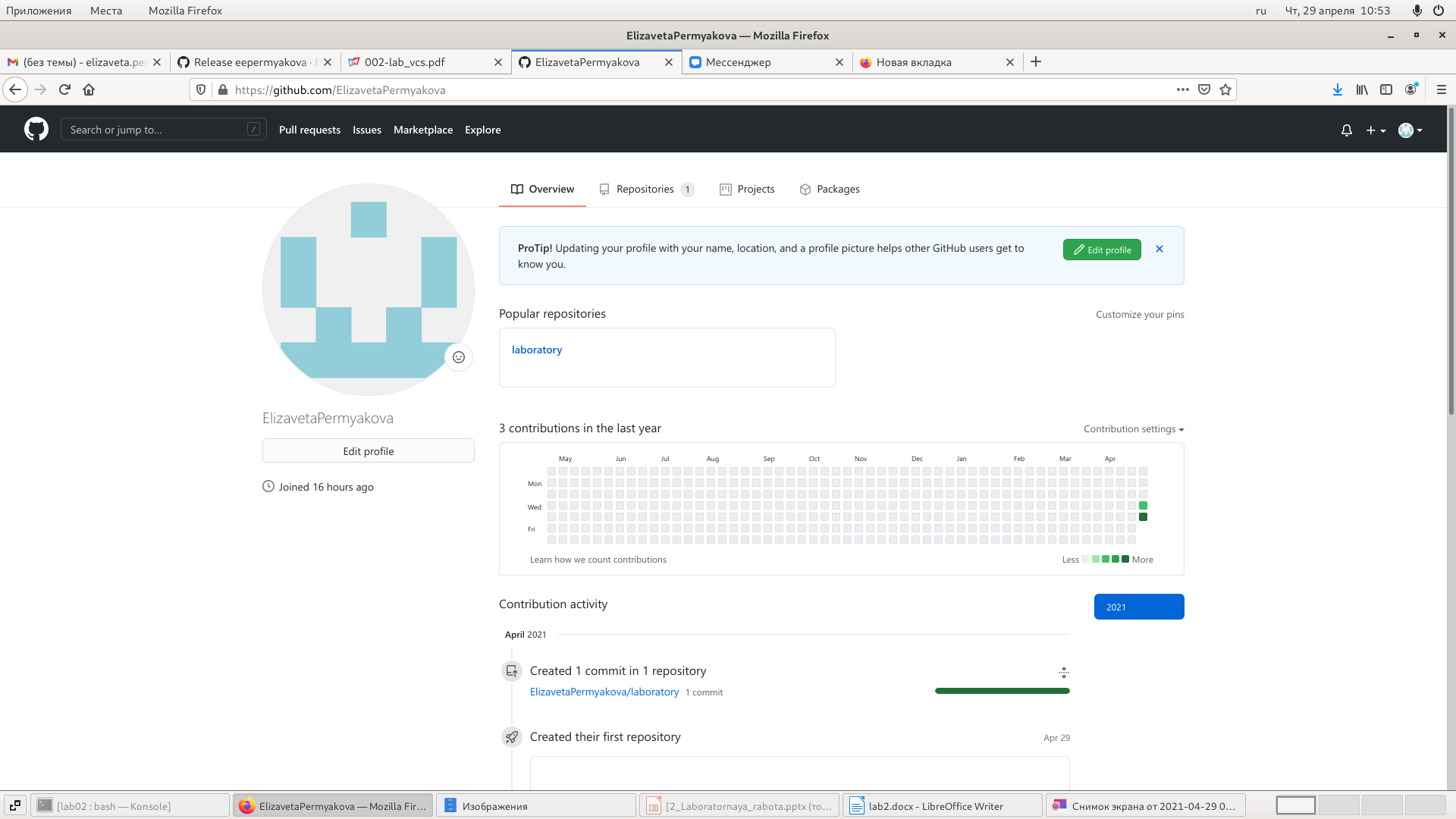
Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

# Задание

Сделайте отчёт по предыдущей лабораторной работе в формате Markdown. В качестве отчёта просьба предоставить отчёты в 3 форматах:pdf,docxиmd(вархиве,поскольку он должен содержать скриншоты,Makefile ит.д.)

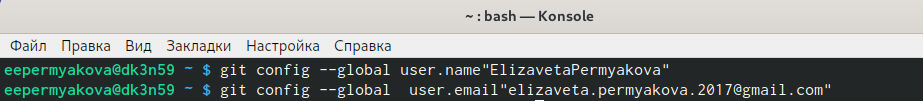
# Выполнение лабораторной работы

Создаем учетную запись на https://github.com.(рис. @fig:001)



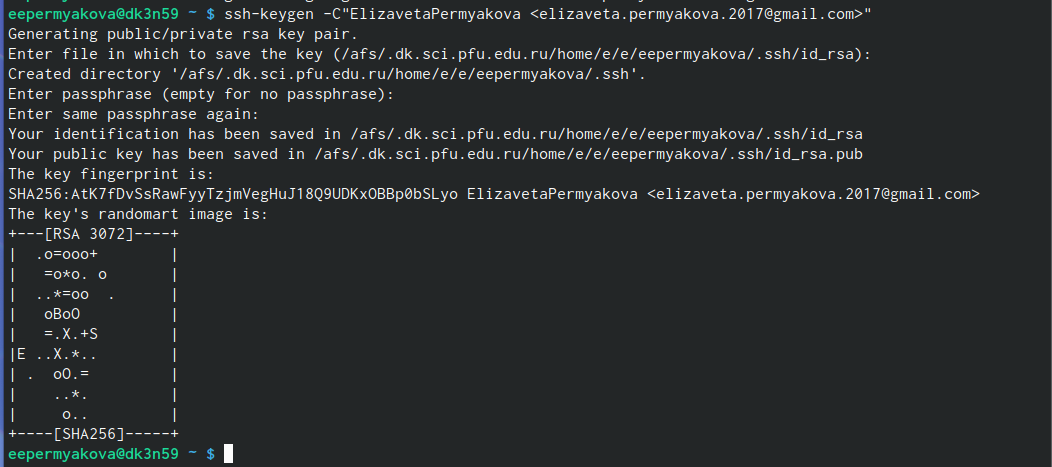
Учетная запись

Настраиваем систему контроля версий git. Синхранизируем учётную запись github с компьютером.(рис. @fig:002)



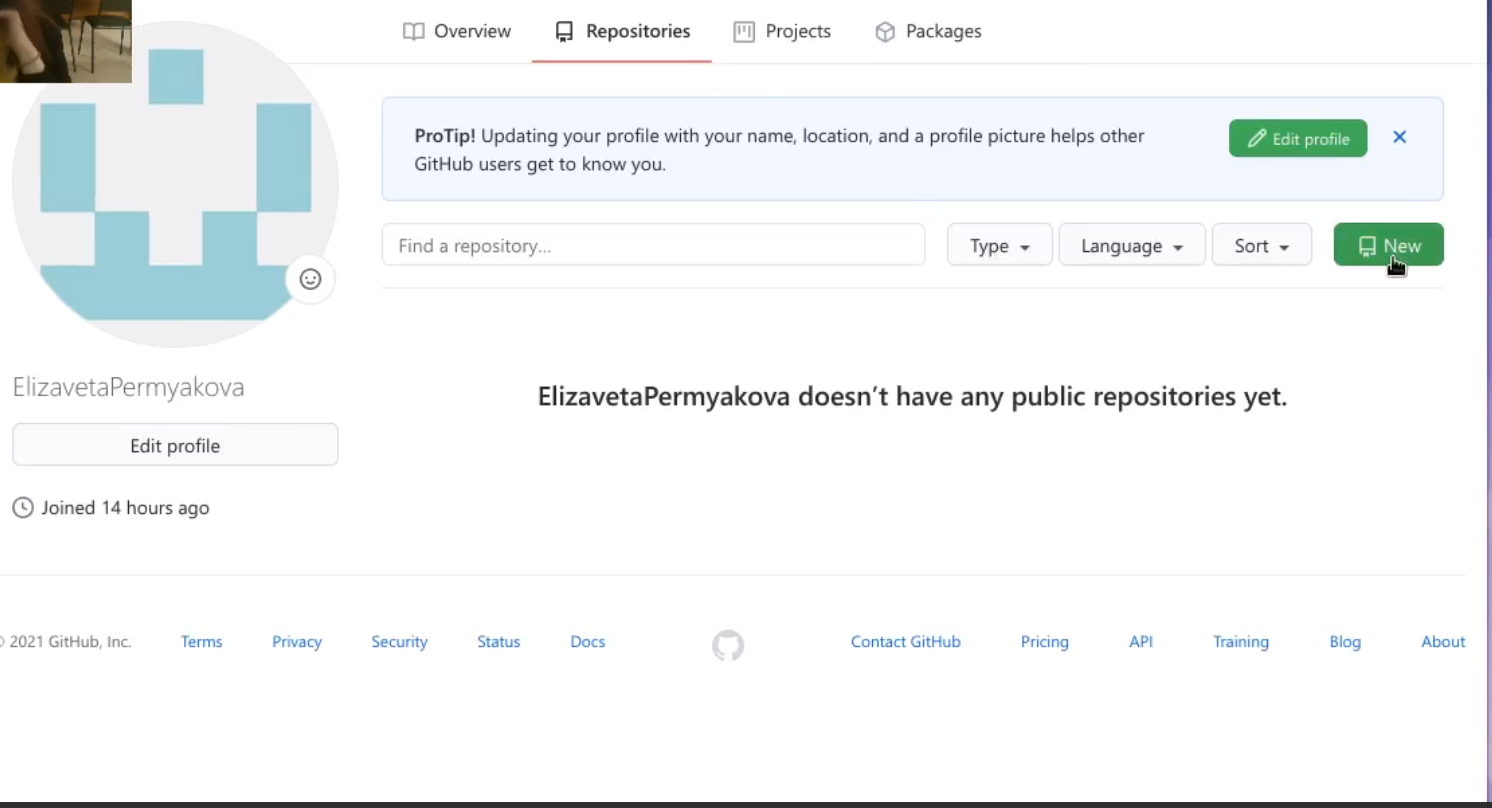
Система контроля git

После этого создаём новый ключ на github и привязываем его к копьютеру через консоль.(рис. @fig:003)



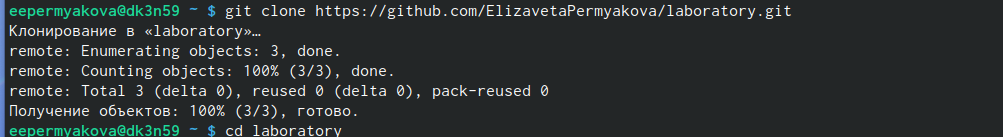
Новый ключ

Следующим шагом будет создание и подключение репозитория к github. В gethup заходим в «repository» и создаём новый репозиторий (имя «laboratory», а заголовок для файла README). Копируем в консоль ссылку на репозиторий (для дальнейшей работы с файлами).(рис. @fig:004)



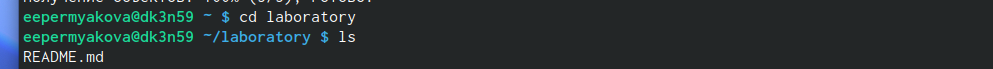
Создание репозитория

(рис. @fig:005)



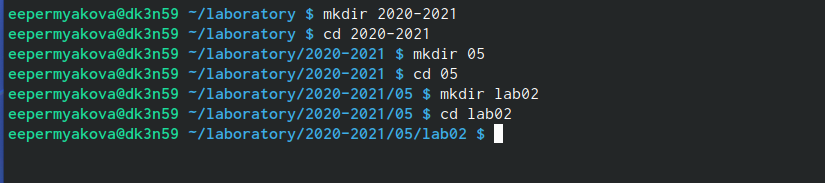
Копируем ссылку на репозиторий

В лабораторной работе описан логаритм создания структуры католога через консоль. Но легче будет создать репозиторий в gethup и после этого работать с каталогом и папками через консоль (перед этим необходимо скопировать ссылку на репозиторий в консоль, в формате https или ssh). Перед тем, как создавать файлы, заходим в наш репозиорий:(рис. @fig:006)



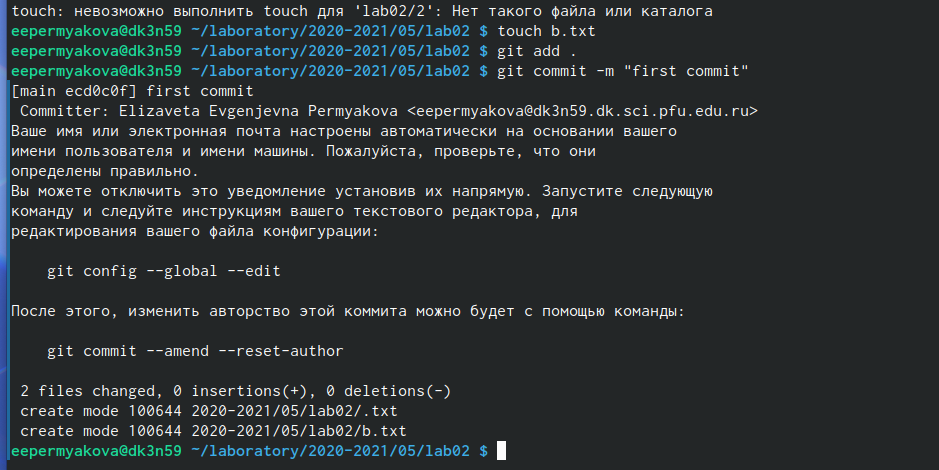
Репозиторий

После этого можем уже создавать наши файлы.(рис. @fig:007)



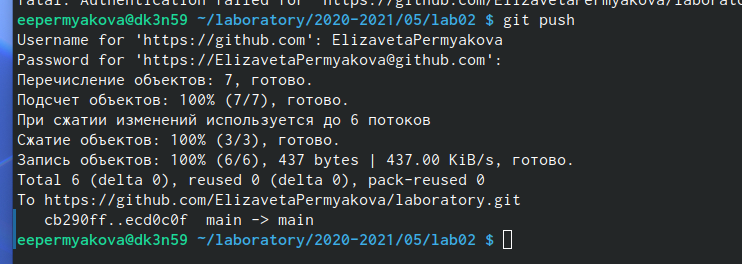
Создание файлов

Добавляем первый коммит и выкладываем на gethup. Для того, чтобы правильно разместить первый коммит, необходимо добавить команду git add ., после этого с помощью команды git commit -m “first commit” выкладываем коммит.(рис. @fig:007)



Первый коммит

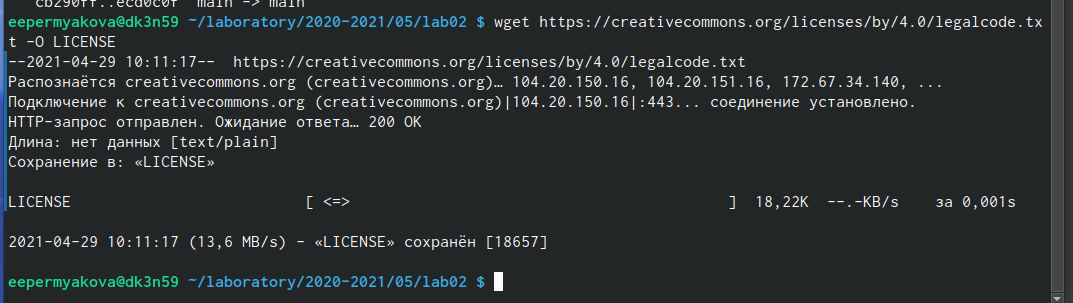
Сохраняем первый коммит, используя команду git push.(рис. @fig:009)



Сохранение

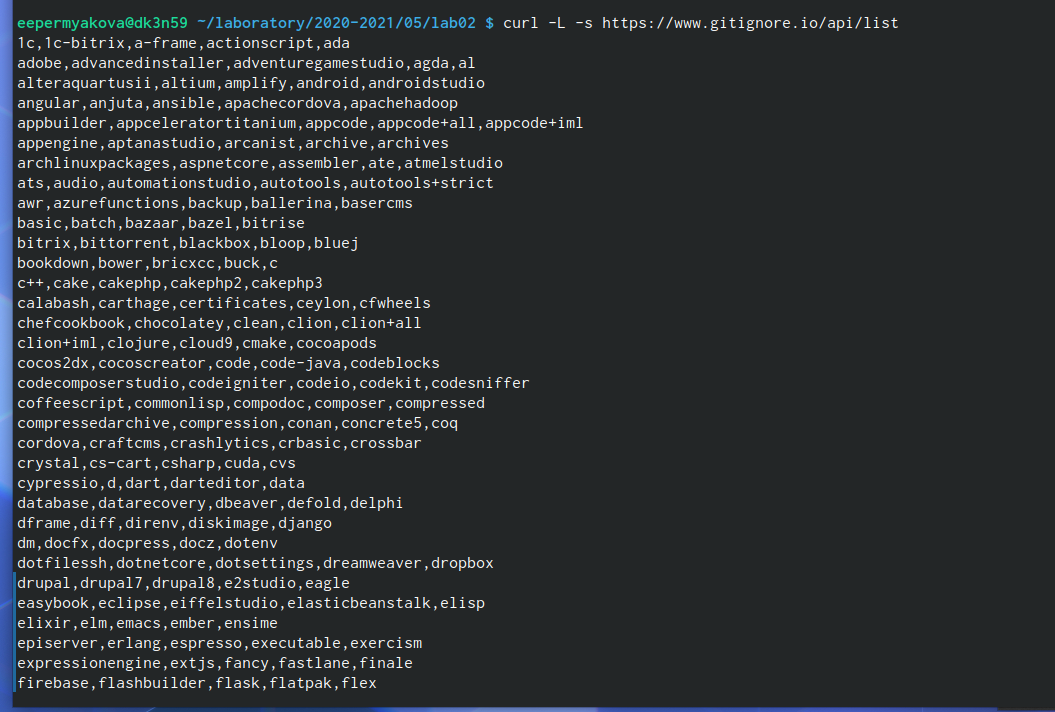
Первичная конфигурация:

1. Добавляем файл лицензии.(рис. @fig:010)



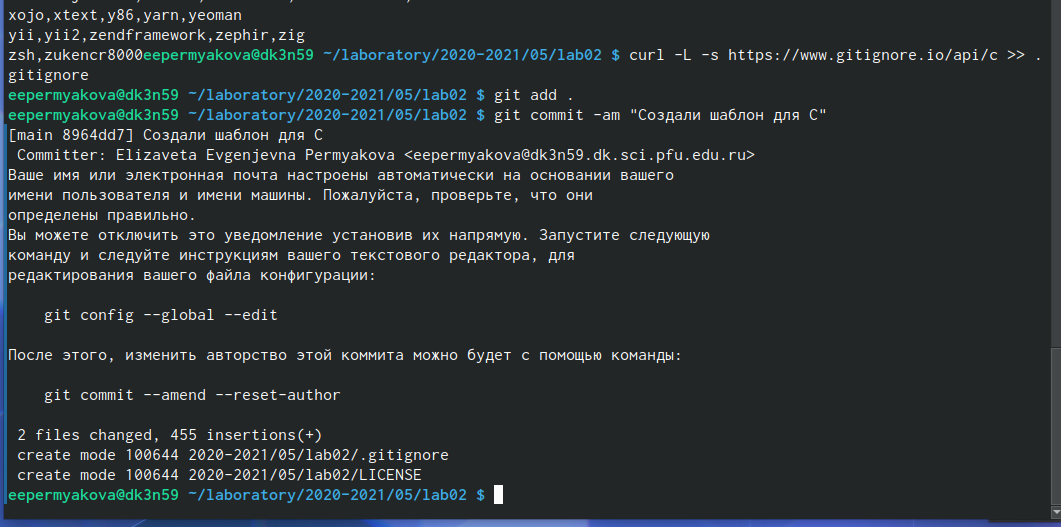
Файл лицензии

1. Добавим шаблон игнорируемых файлов. Просмотрим список имеющихся шаблонов (на скриншоте список шаблонов представлен не в целом виде).(рис. @fig:011)



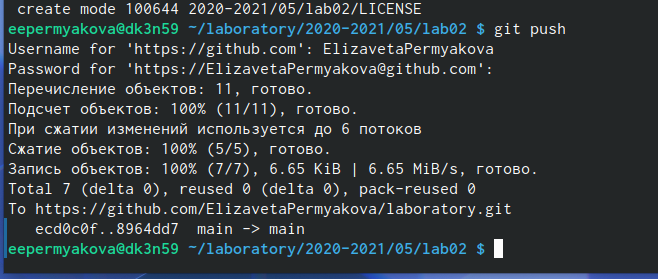
Шаблон игнорируемых файлов

1. Скачиваем шаблон, например, для C. Также добавляем новые файлы и выполняем коммит.(рис. @fig:012)



Скачивание шаблона

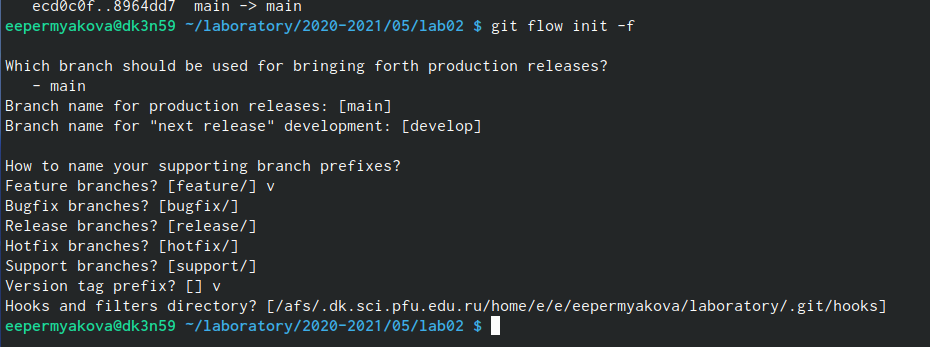
1. Отправим на github (для этого сохраним все созданные шаблоны и файлы, используя команду git push).(рис. @fig:013)



Отправление на github

Работаем с конфигурацией git-flow.

1. Инициализируем git-flow, используя команду git flow init -f (префикс для ярлыков установлен в v).(рис. @fig:014)



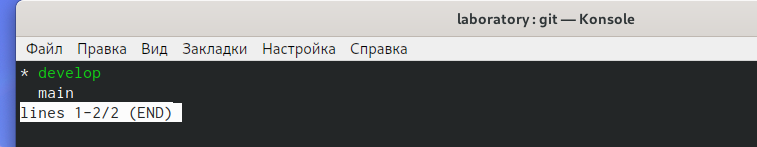
Инициализация git-flow

1. Проверяем, что мы находимся на ветке develop (используем команду git branch).(рис. @fig:015)

git brunch

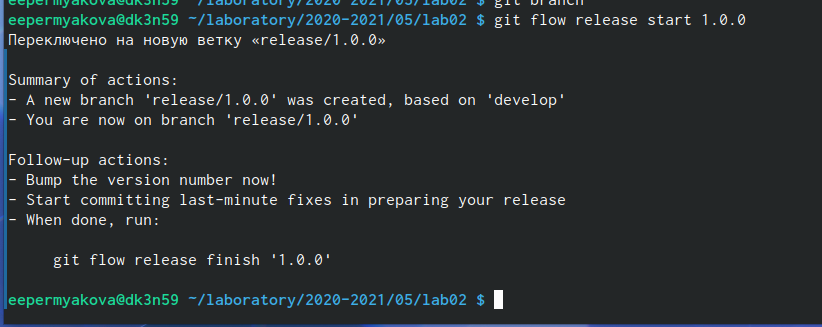
git brunch

(рис. @fig:016)



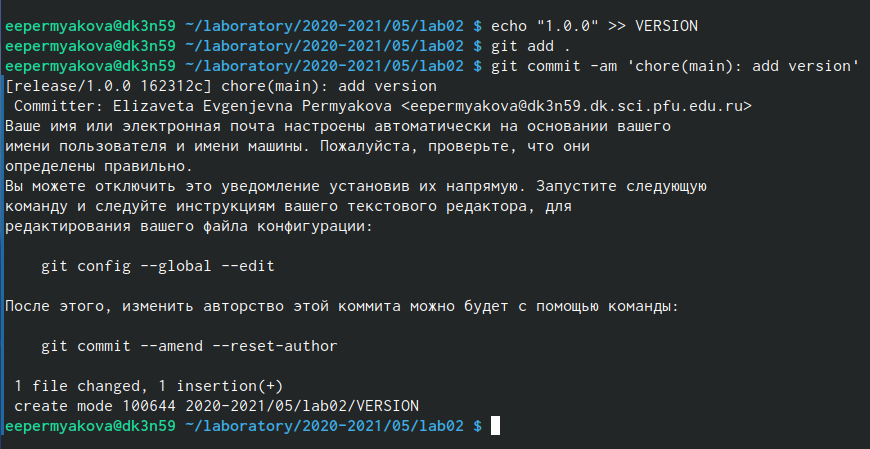
Ветка develop

1. Создаём релиз с версией 1.0.0.(рис. @fig:017)



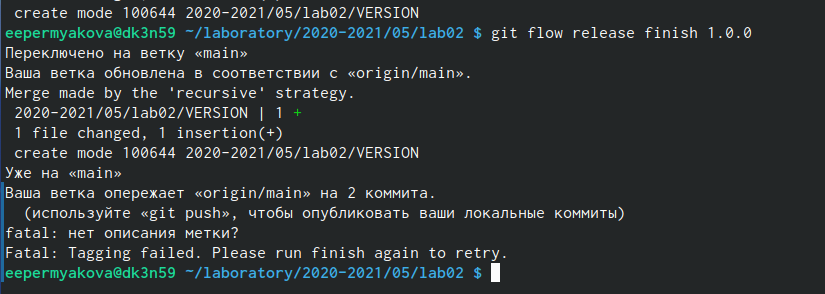
Создание релиза

1. Запишем версию и добавим в индекс.(рис. @fig:018)



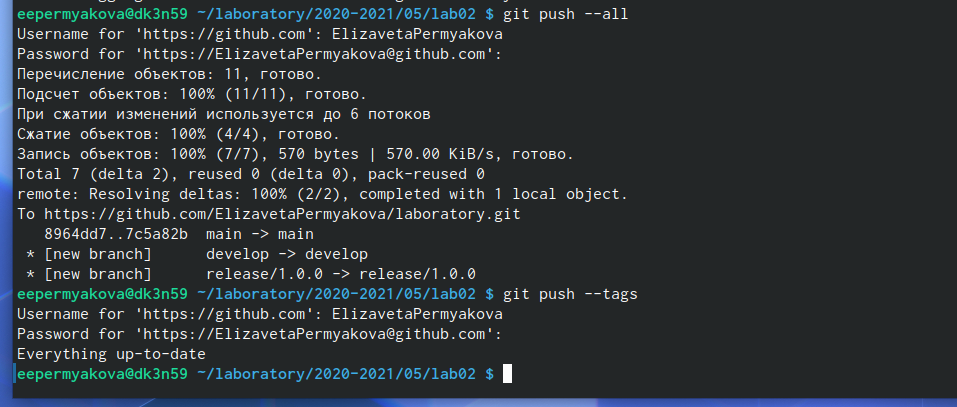
Версия

1. Заливаем релизную ветку в основную ветку (используем команду git flow release finish1.0.0).(рис. @fig:019)



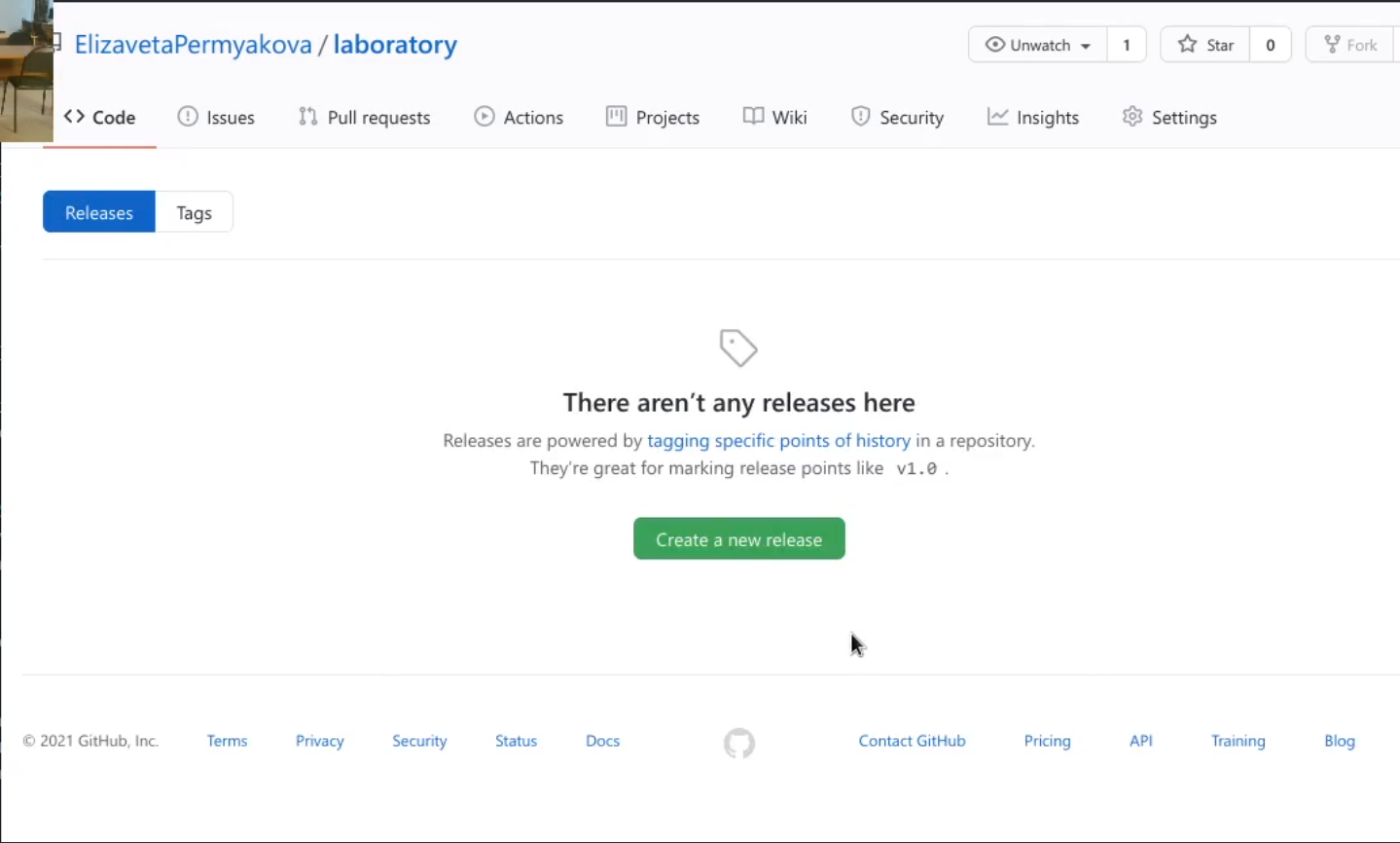
Релизная ветка

1. Отправляем данные на github.(рис. @fig:020)



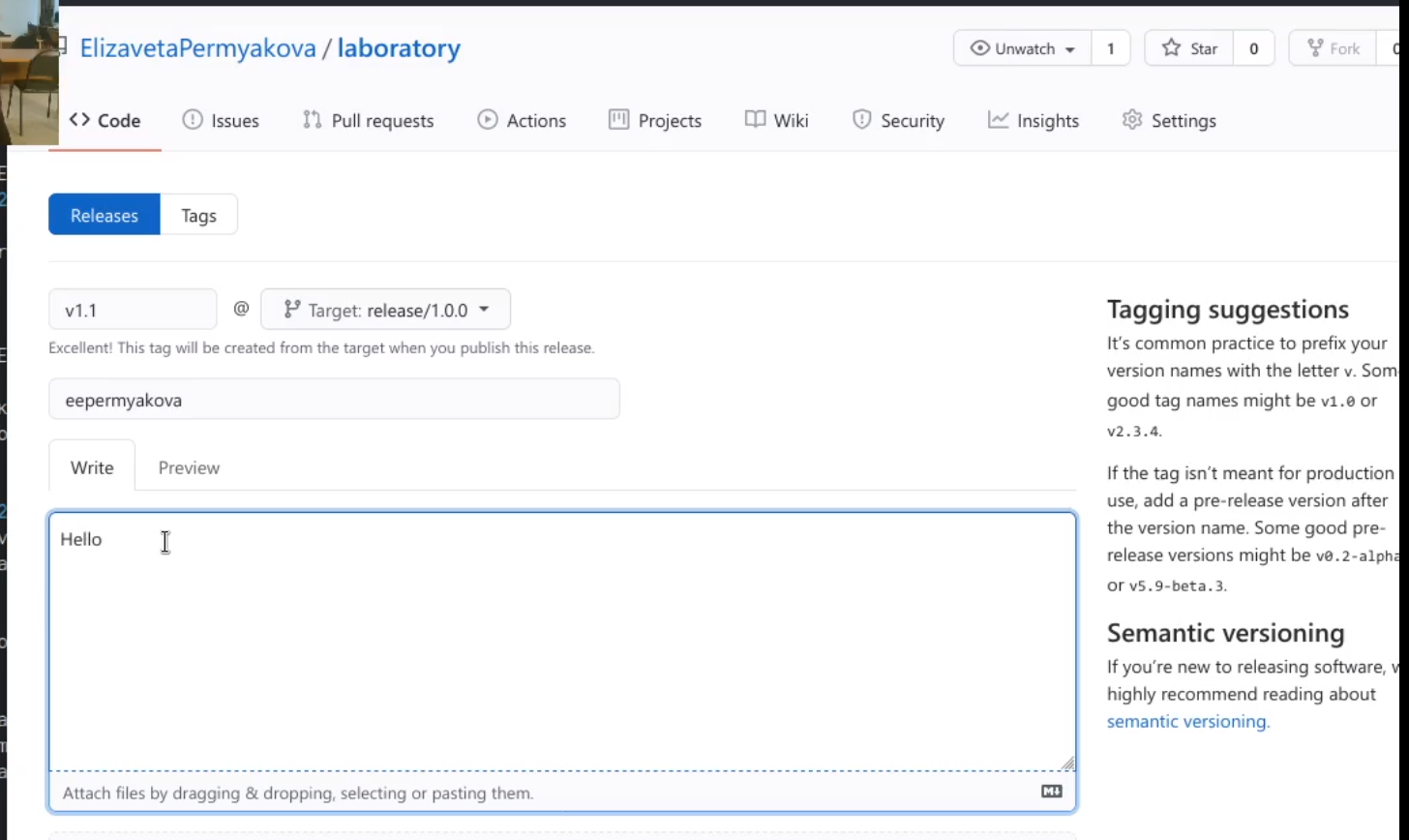
Отправление данных

Создаем релиз на github. Для этого заходим в «Releases», нажимаем «Создать новый релиз». Заходим в теги и заполняем все поля (создаём теги для версии 1.0.0). После создания тега, автоматически сформируется релиз.(рис. @fig:021)



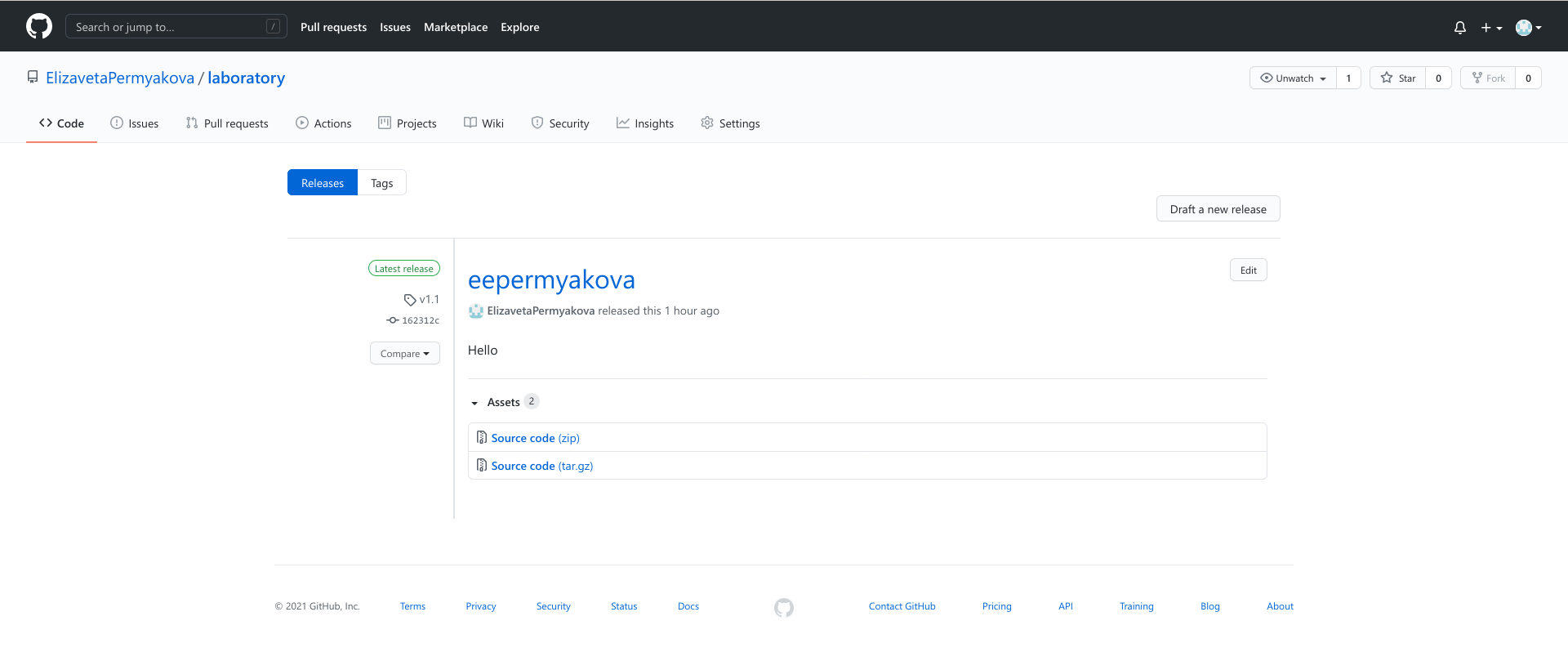
Releases

(рис. @fig:022)



Заполнение полей

(рис. @fig:023)



Сформированный релиз

# Выводы

Я изучила идеологию и научилась применять средства контроля версий.