# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

**Факультет Физико-Математических Наук Кафедра прикладной информатики и теории вероятностей**

# ОТЧЁТ

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2**

*дисциплина: Операционные системы*

Студент: Евдокимов Иван Андреевич Группа: НФИбд-01-20

Преподаватель: Кулябов Дмитрий Сергеевич

# МОСКВА

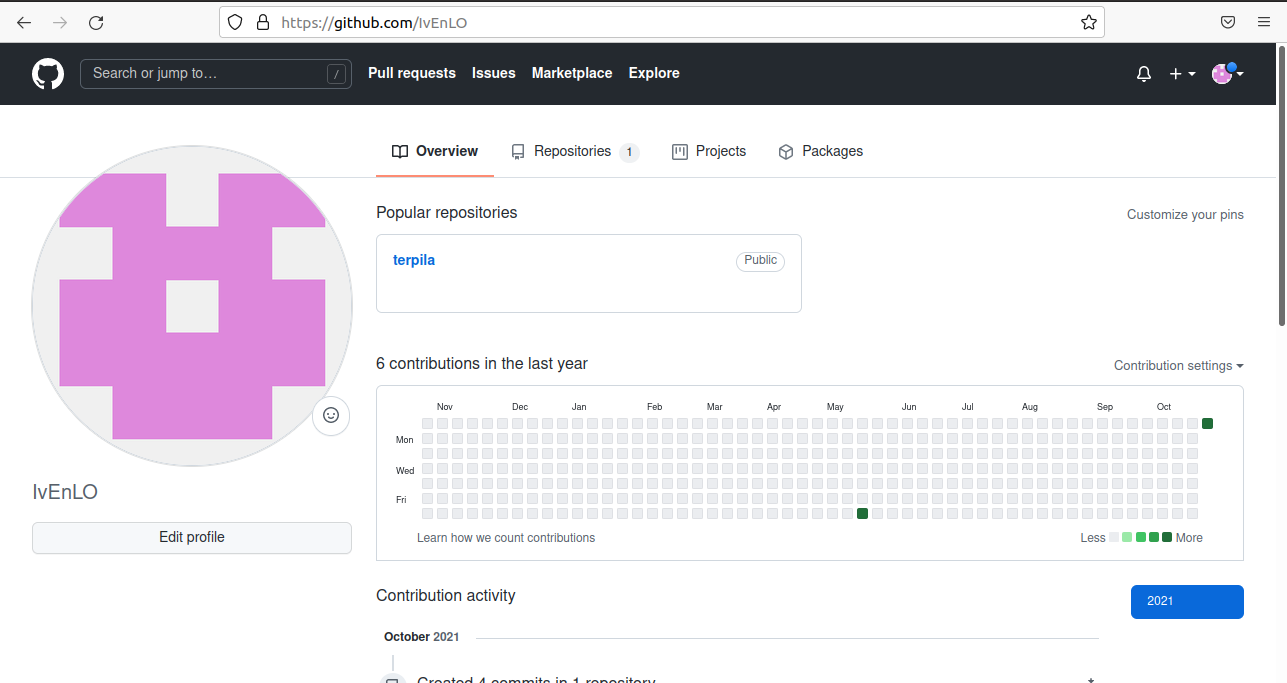
2021 г.

**Цель работы:** Изучить идеологию и применение средств контроля версий

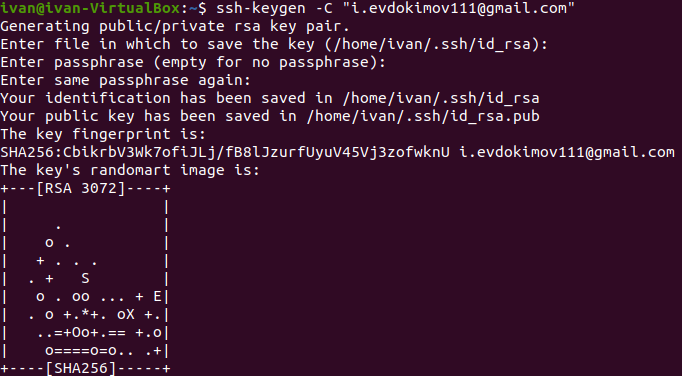
# Этапы работы:

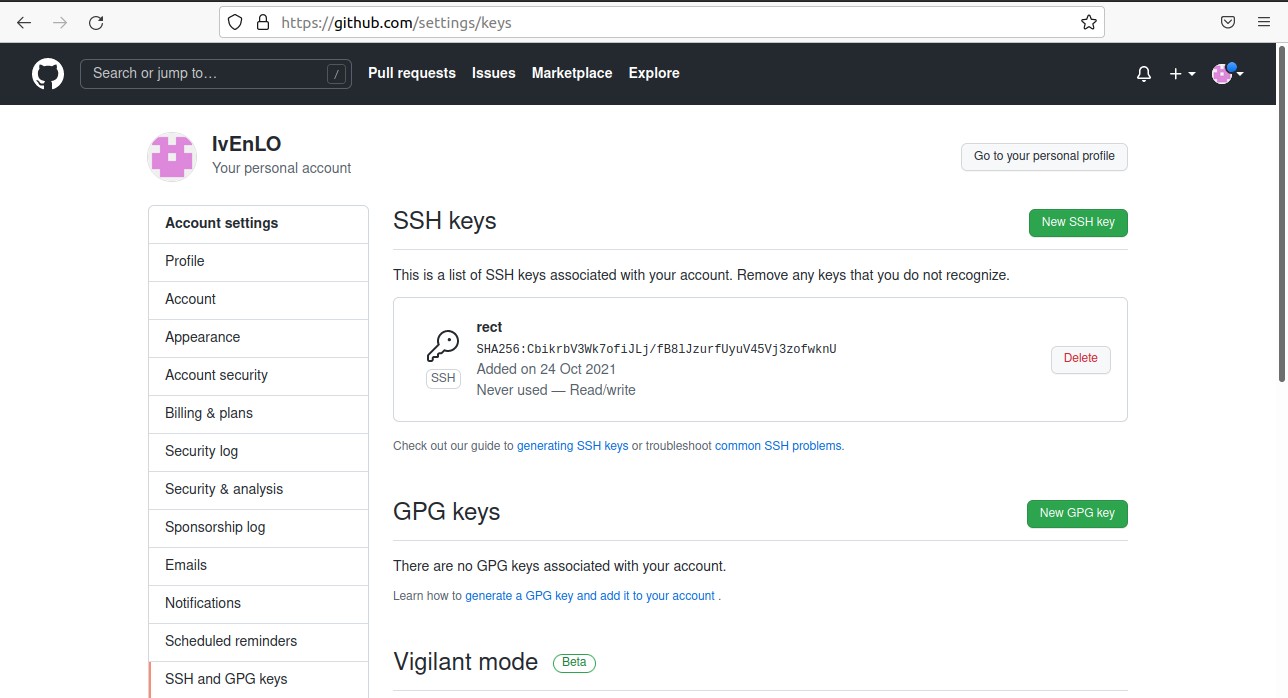
*1.1 Настройка git*

Создадим учётную запись на https://github.com.

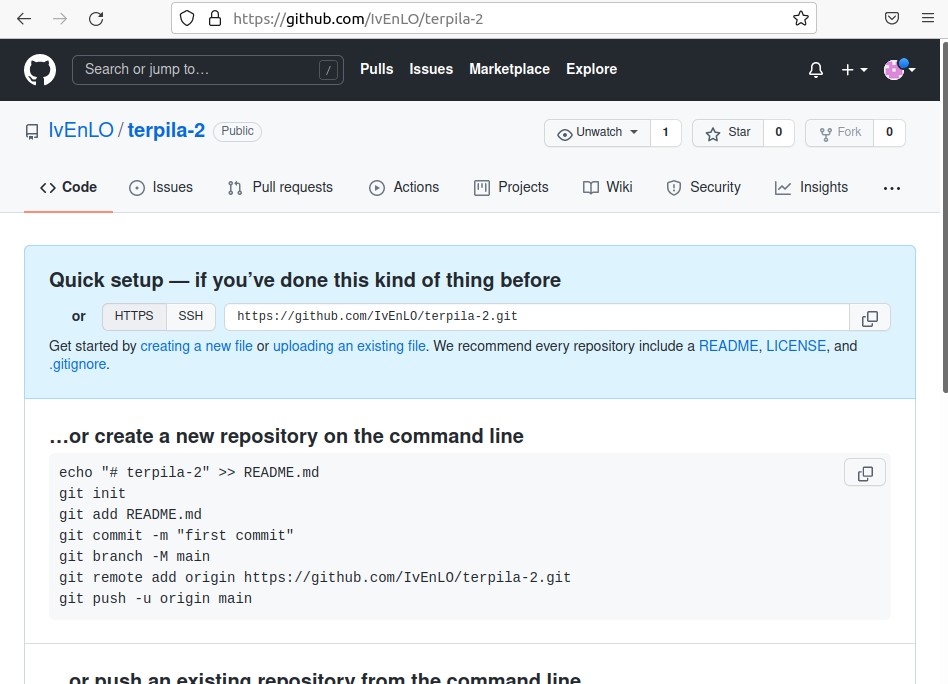


Настроим систему контроля версий git, как это описано выше c использованием сервера репозиториев https://github.com/.





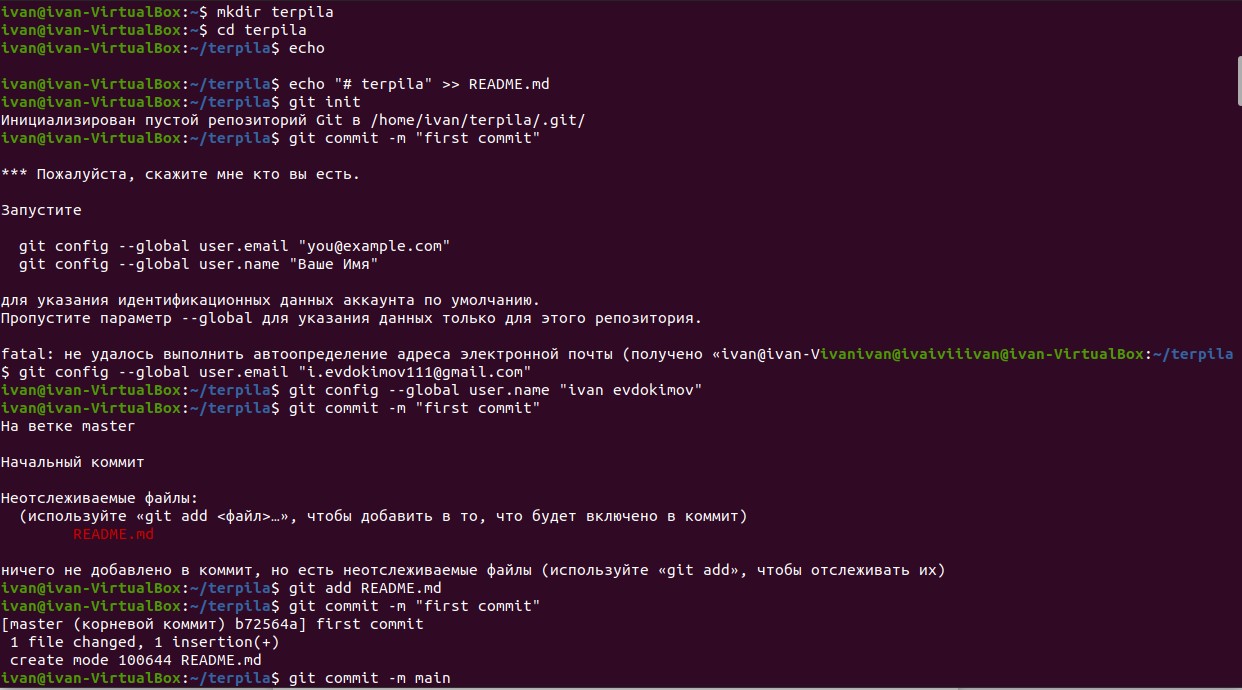
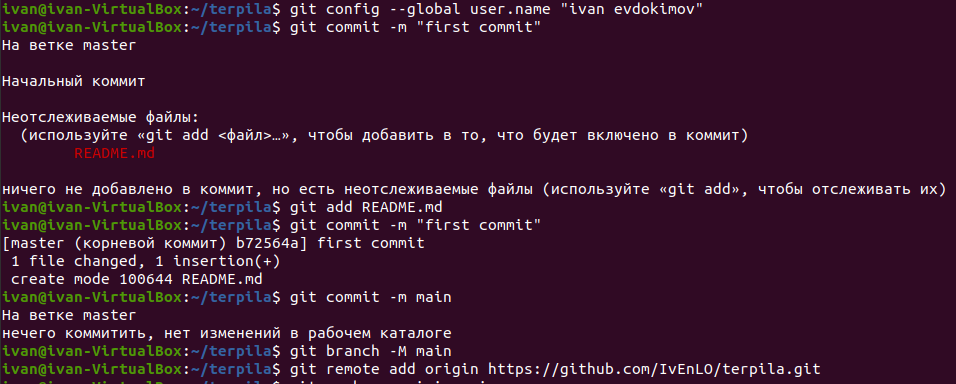
Создадим структуру каталога лабораторных работ согласно пункту М.2



* 1. *Подключение репозитория к github* Создадим репозиторий на GitHub. Для примера назовём его laboratory. • Рабочий каталог будем

обозначать как laboratory. Вначале нужно перейти в этот каталог: cd laboratory • Инициализируем системы git: git init • Создаём заготовку для файла README.md: echo ”# Лабораторные работы” >> README.md • Делаем первый коммит и выкладываем на github

git add README.md git commit -m ”first commit” git remote add origin git@github.com:LLIAJIYH/laboratory.git git push -u origin master

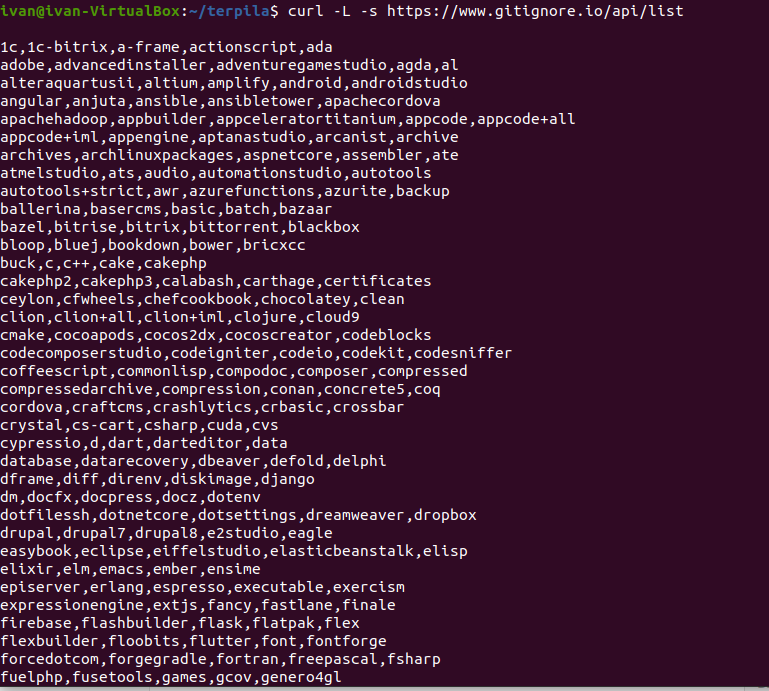


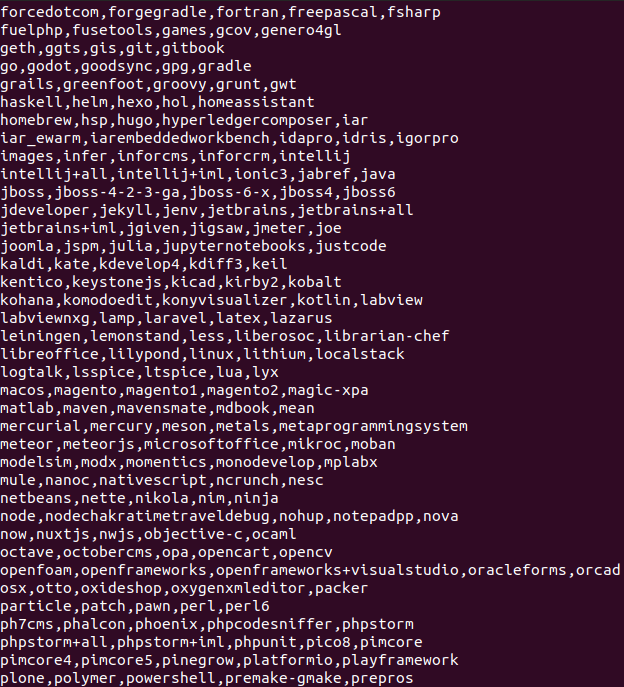


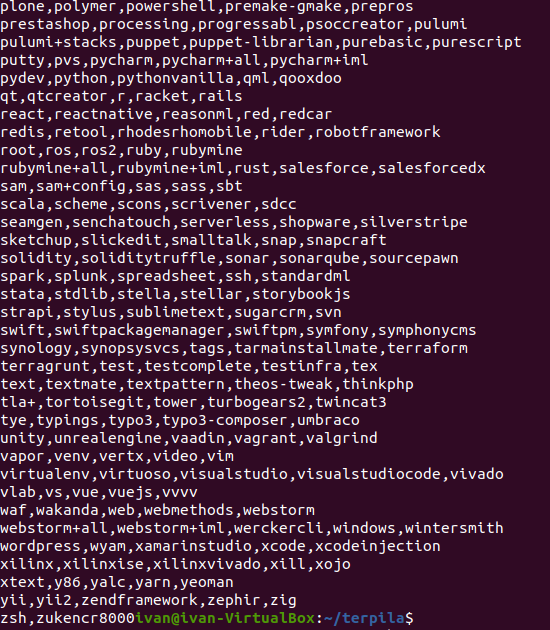
* 1. *Первичная конфигурация*
* Добавим файл лицензии: wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE • Добавим шаблон игнорируемых файлов. Просмотрим список имеющихся шаблонов: curl -L -s https://[www.gitignore.io/api/list](http://www.gitignore.io/api/list)



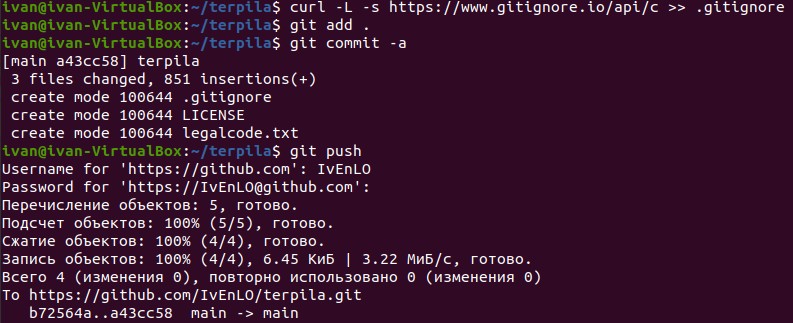
Затем скачаем шаблон, например, для C: curl -L -s https://[www.gitignore.io/api/c](http://www.gitignore.io/api/c) >> .gitignore







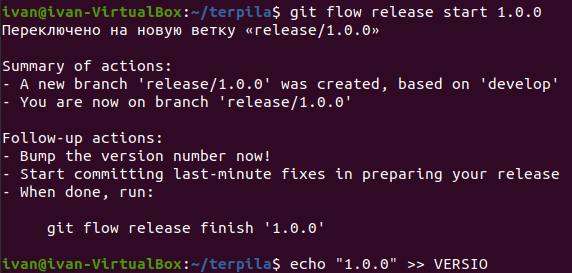
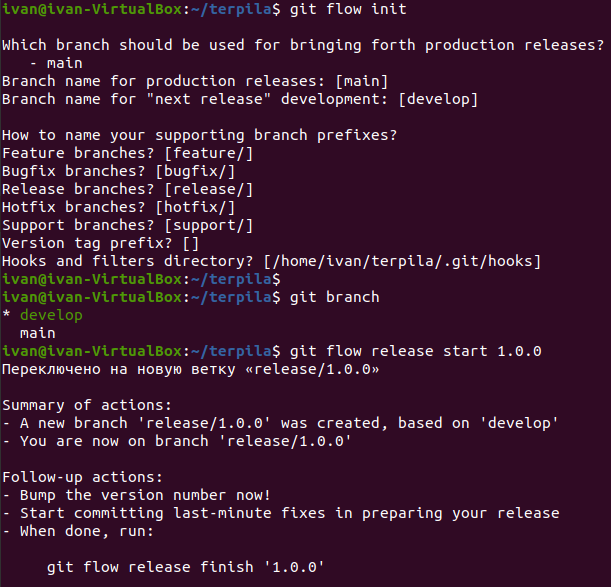
* Добавим новые файлы: git add .
* Выполним коммит: git commit -a git push



* 1. *Конфигурация git-flow*
* Инициализируем git-flow git flow init
* Префикс для ярлыков установим в v.
* Проверим, что Вы на ветке develop: git branch

Создадим релиз с версией 1.0.0

git flow release start 1.0.0



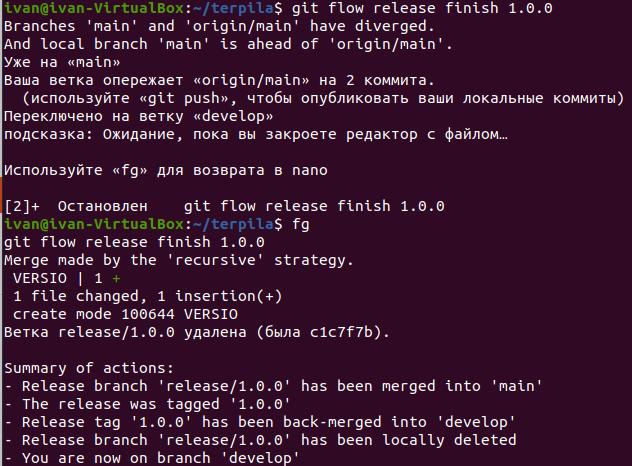
Запишем версию:

echo ”1.0.0” >> VERSION

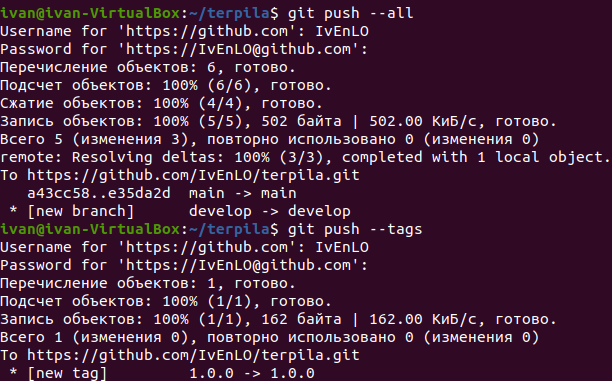
Добавим в индекс:

git add .

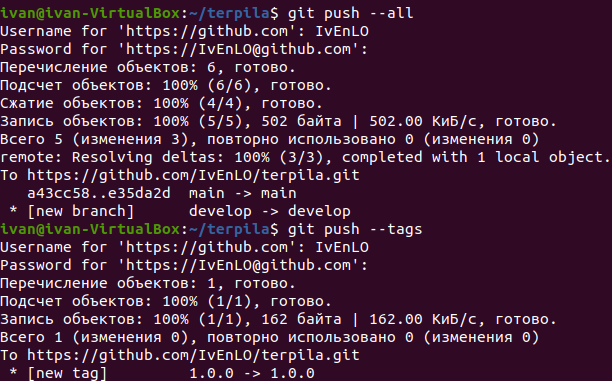
git commit -am 'chore(main): add version' Зальём релизную ветку в основную ветку git flow release finish 1.0.0

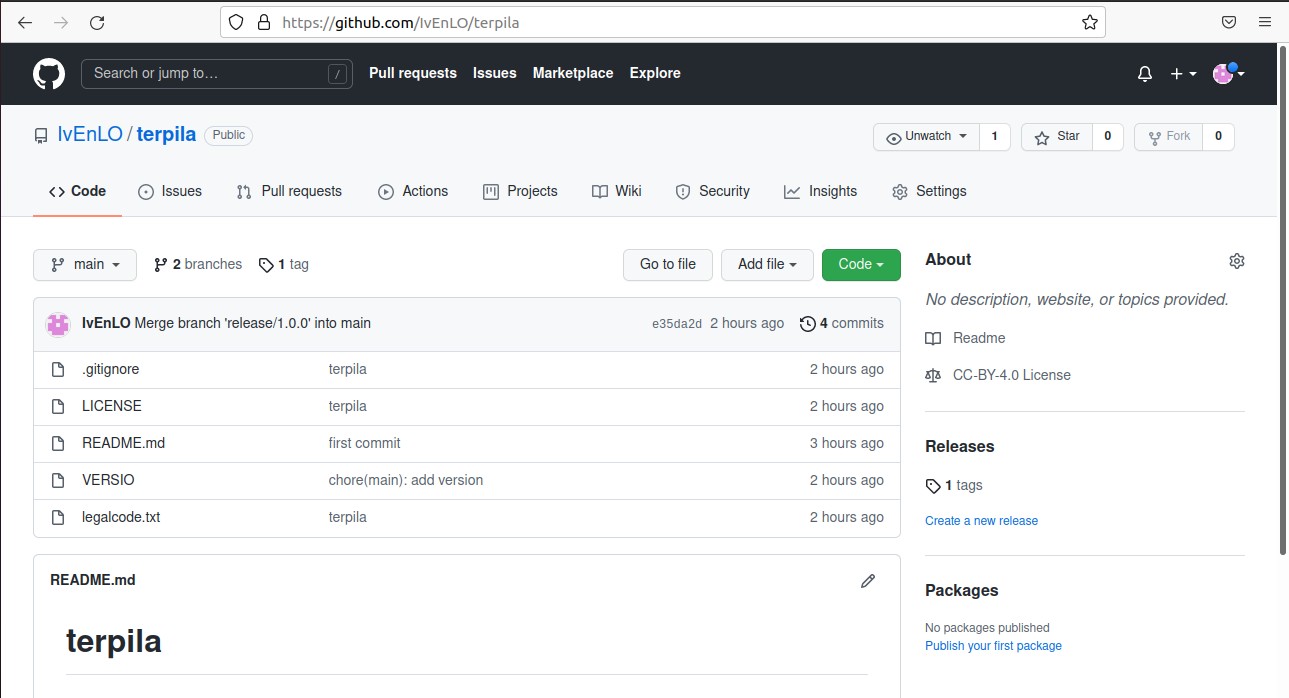


Отправим данные на github git push --all



Создадим релиз на github git push --tags





**Выводы:** По мере выполнения данной работы я изучил идеологию и применение средств контроля версий

# Контрольные вопросы:

*1)Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?*

Ответ: Система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

\*2) Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. \*

Ответ:

* некоторое место, где хранятся файлы
* commit - отдельная версия программы
* история - информация о всех изменениях в проекте
* рабочая копия - текущее состояние файлов проекта

\*3) Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. \*

Ответ: децентрализованные хранятся на многих устройствах и передаются многим устройствам(Mercurial), централизованные - файлы хранятся в одном устройства(CVS)

\*4)Опишите действия с VCS при единоличной работе с хранилищем. \*

Ответ: изменяем, делаем коммиты, закидываем на репозиторий, если необходимо делаеи релизы.

*5) Опишите порядок работы с общим хранилищем VCS.*

Ответ: делаем пул нынешних изменений, изменяем, добавляем свои изменения в ветку, коммиты и т.д

\*6)Каковы основные задачи, решаемые инструментальным средством git? \*

Ответ: контроль версиями, одновременная работа нескольких людей над 1 проектом и многое другое

\*7)Назовите и дайте краткую характеристику командам git. \*

Ответ: -init = инициализация локального репозитория -status = показ всех измененных и добавленных в коммит файлов -add = добавление файлов

\*8)Приведите примеры использования при работе с локальным и удалённым репозиториями. \*

Ответ: проект с закрытым исходным кодом используют локальные репозитории, проекты с открытым исходным кодом, проекты выполняемые несколькими людьми, удаленно - удаленные репозитории

\*9)Что такое и зачем могут быть нужны ветви (branches)? \*

Ответ: ветки используются для удобства работы с версиями и изминениями программы

\*10)Как и зачем можно игнорировать некоторые файлы при commit? \*

Ответ: многие конфигурационные файлы с личными или важными файлами не должны быть доступны, а также есть примеры папок/файлов, которые создаются для кэширования, которые независимы от проекта и от которых проект не зависит.