**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**ОТЧЕТ**

**по лабораторной работе № 2**

*дисциплина: Операционные системы*

Студент: Оразгелдиева Огулнур

Группа: НПИбд-02-20

**МОСКВА**

2021 г.

**Лабораторная работа №2**

**Управление версиями**

**Цель работы:**

изучить идеологию и применение средств контроля версий

**Ход работы:**

1. Создала учетную запись на github (<https://github.com/oorazgeldiyeva>) (см. рис. 1)

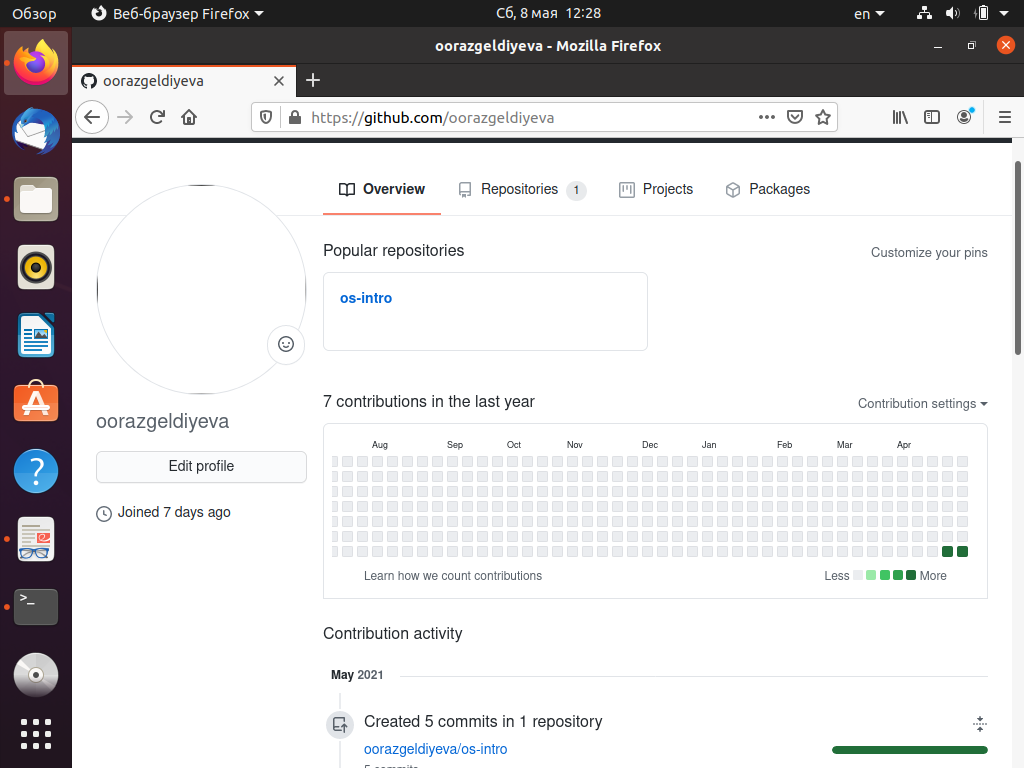


Рисунок 1. Учетная запись на github

1. Сделала конфигурацию с помощью команды git config, потом создала ssh-ключ, и скопировав его в буфер обмена, вставила на github (см. рис. 2-4)

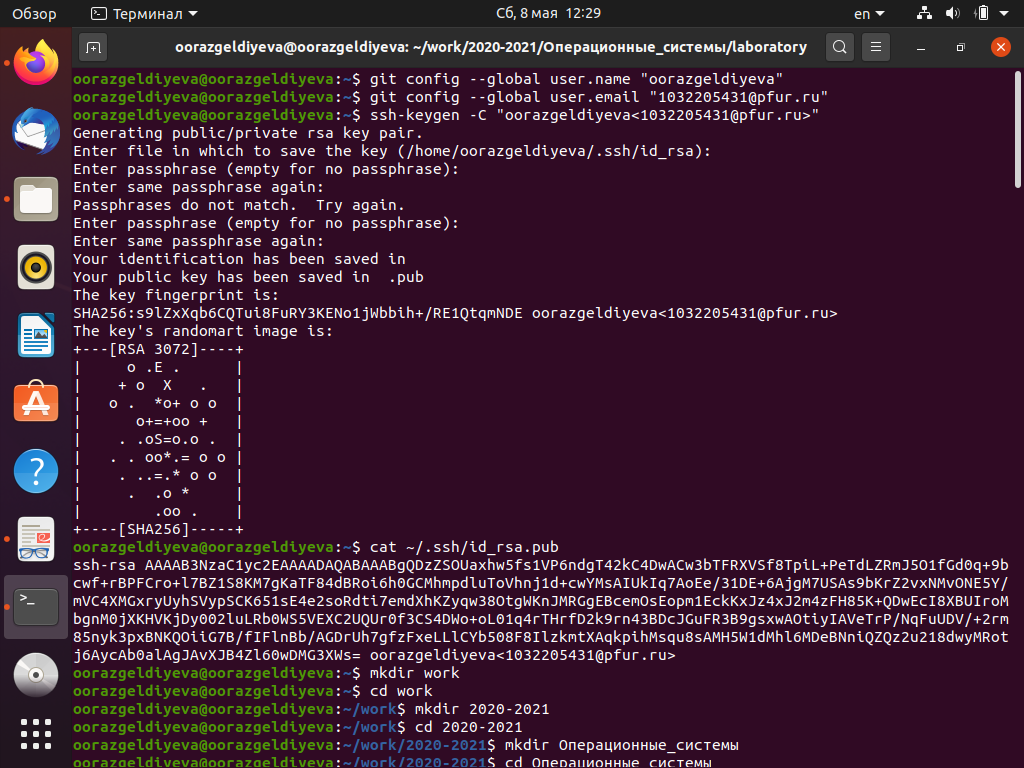


Рисунок 2. Конфигурация github

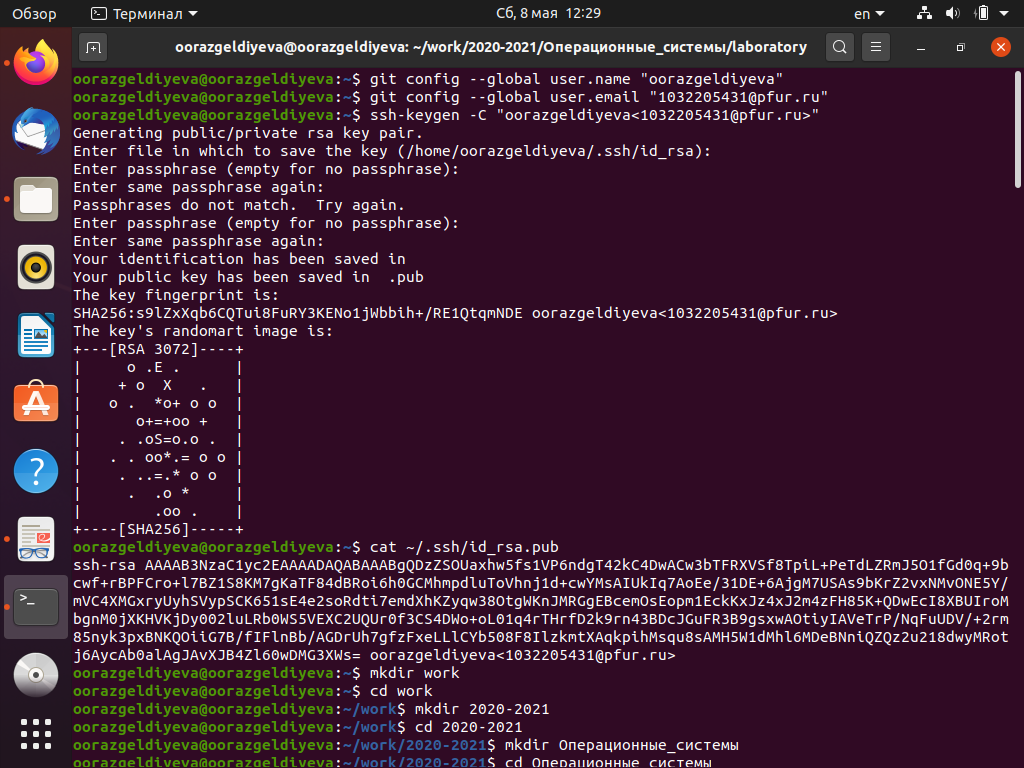


Рисунок 3. Создание и копирование ssh-ключа

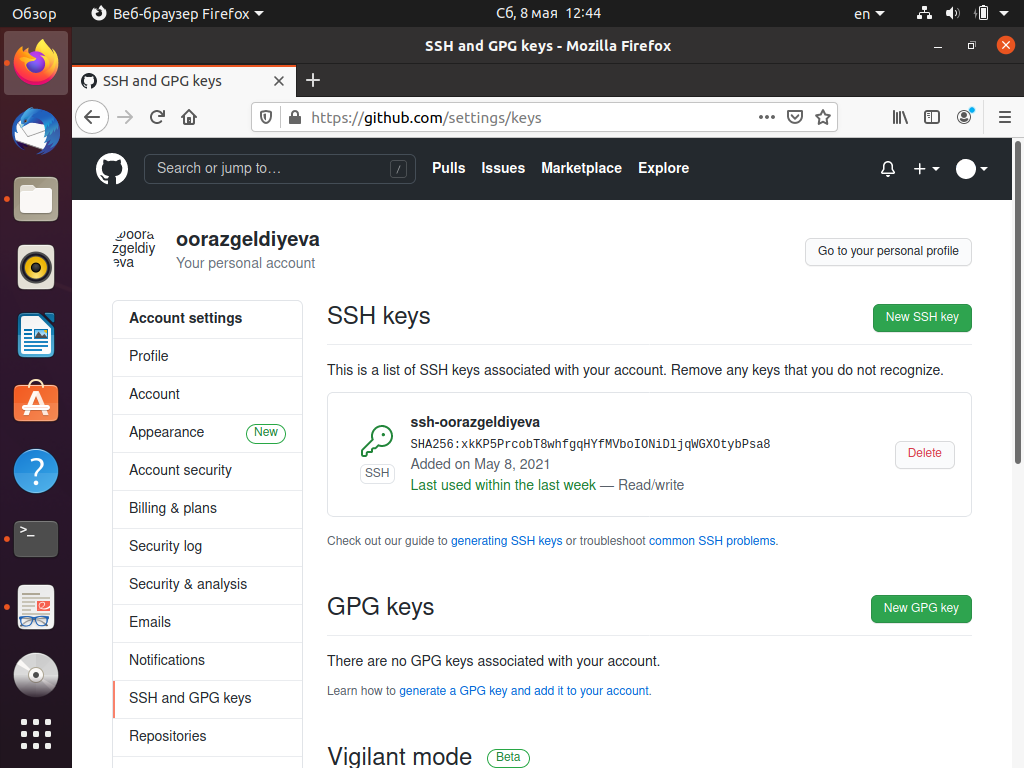


Рисунок 4. Добавление ssh-ключа

1. Создала каталог файлов согласно пункту М2 (см. рис. 5)

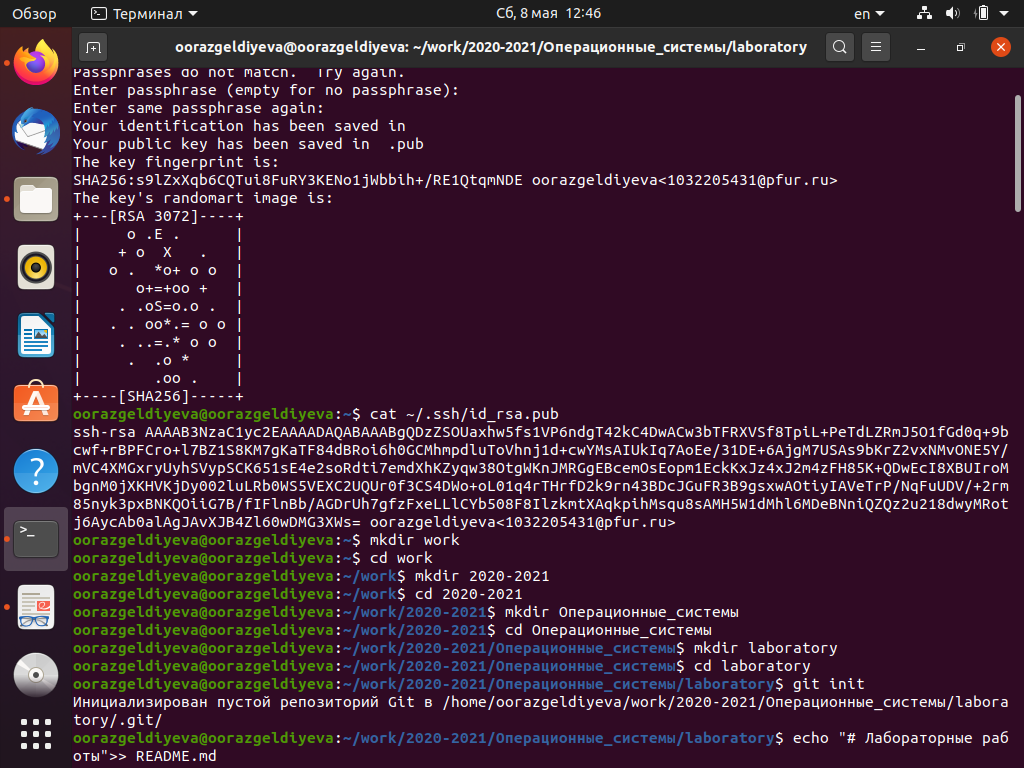


Рисунок 5. Создание каталога

1. Инициализировала git, сделав первый коммит, выложила файл README.md на github. Использовала команды еcho (для создания «загатовки» для коммита), git add (добавление), git commit (для коммита), git push (выложить на гит).

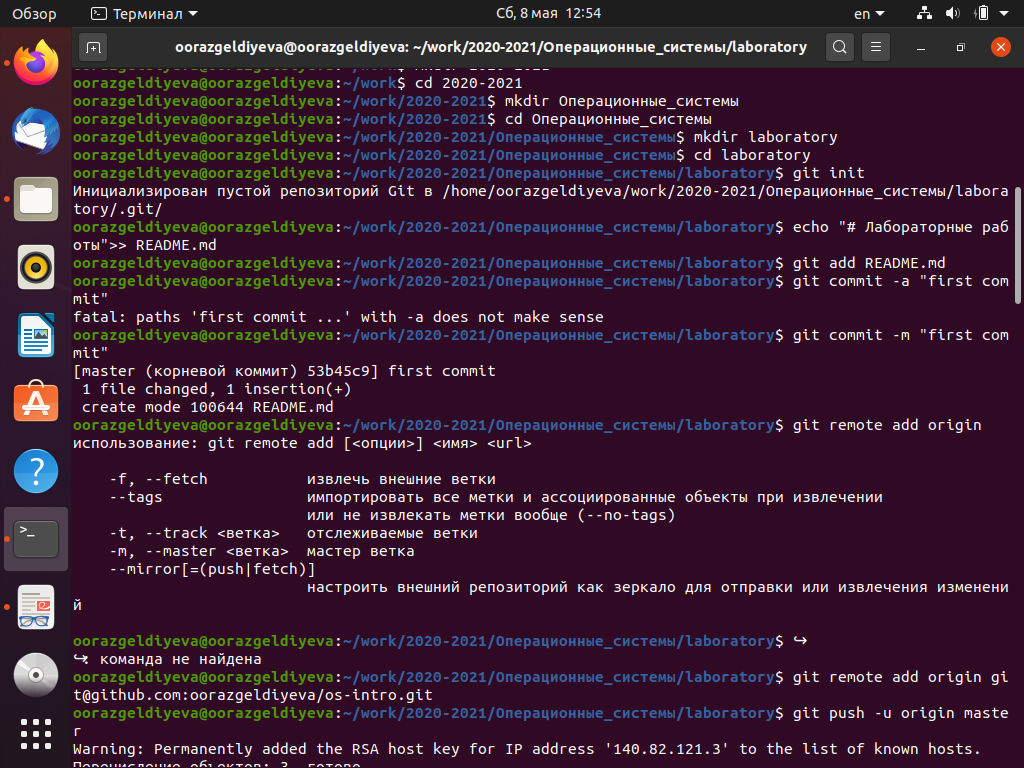


Рисунок 6. Первый коммит

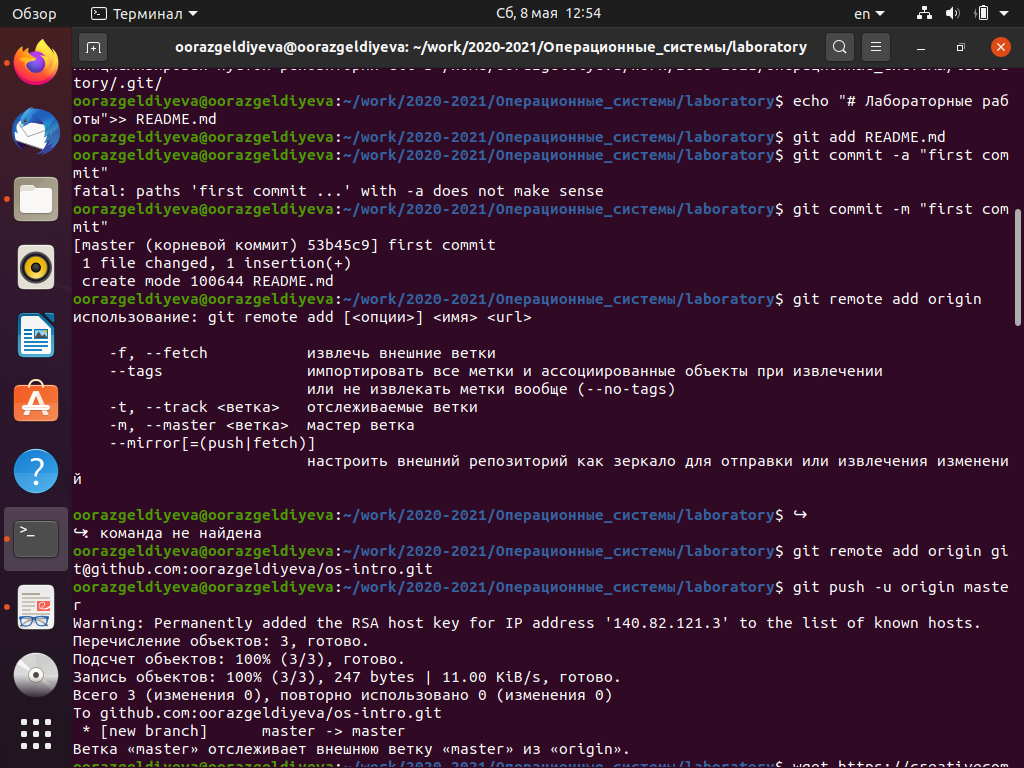


Рисунок 7. Первый коммит

1. Скачала файл лицензии, загрузила шаблон игнорируемых файлов для С и выложила все на github. (см. рис. 8)
2. Сначала установила git-flow с помощью sudo apt-get install git-flow. Инициализировала gitflow (префикс ярлыков установила в v), убедилась, что нахожусь на ветке develop (см. рис. 9)
3. Создала релиз с версией 1.0.0 (см. рис. 10-12)

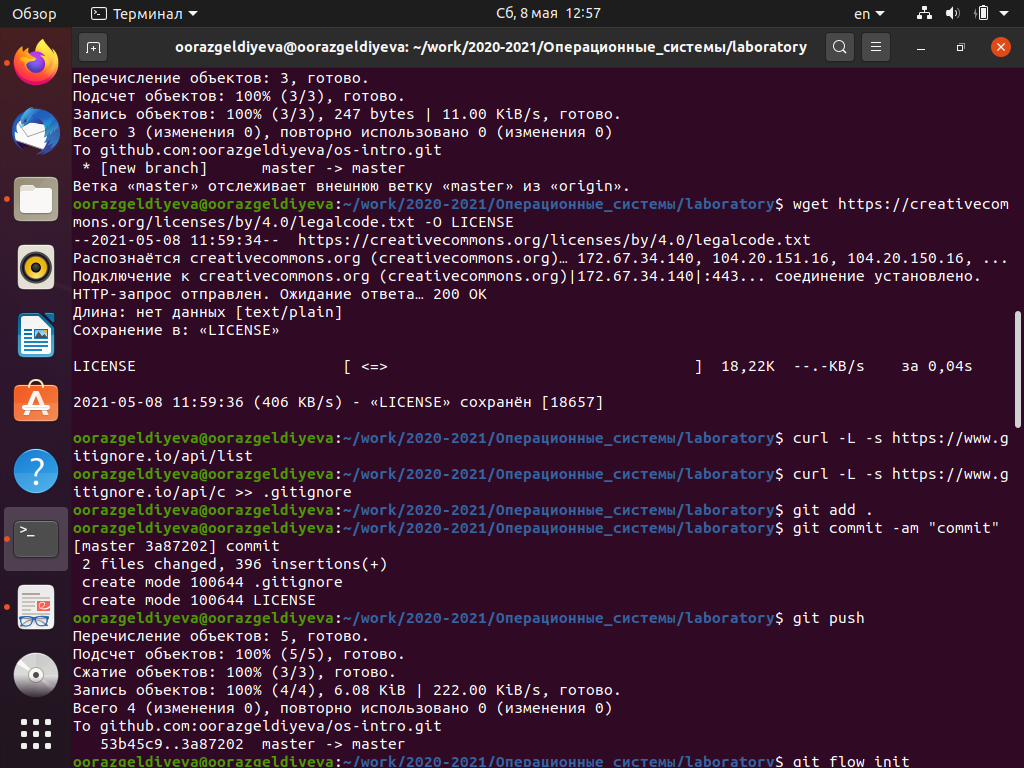


Рисунок 8. Лицензия и игнорируемые файлы

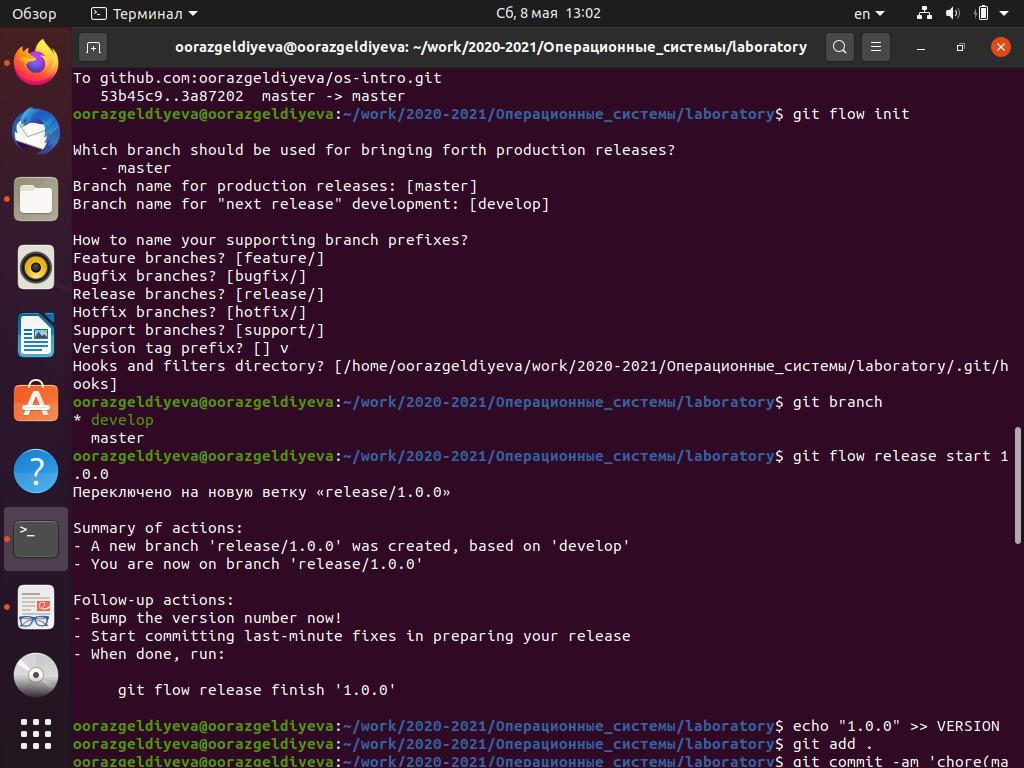


Рисунок 9. Инициализация gitflow

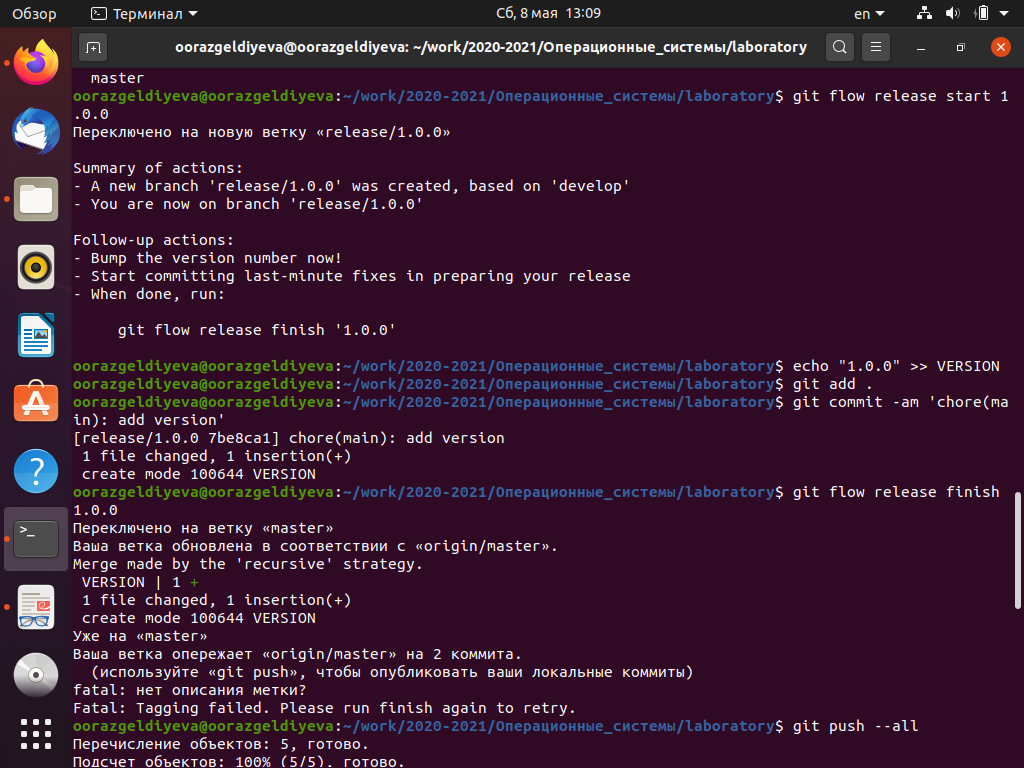


Рисунок 10. Создание релиза версией 1.0.0

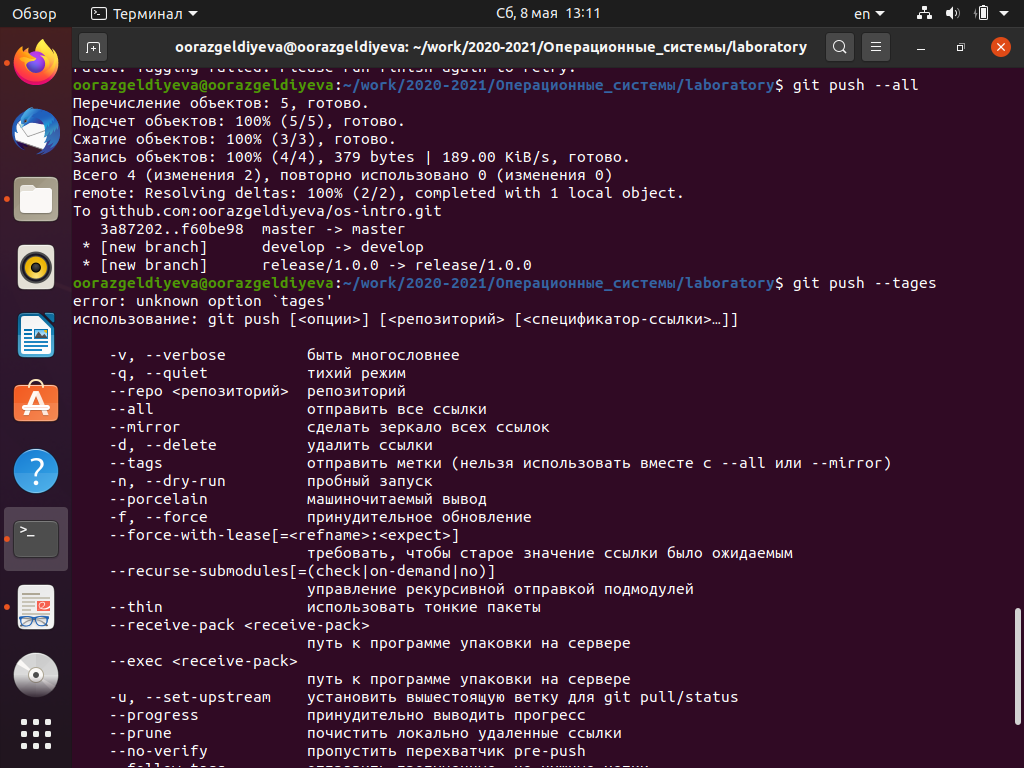


Рисунок 11. Создание релиза версией 1.0.0

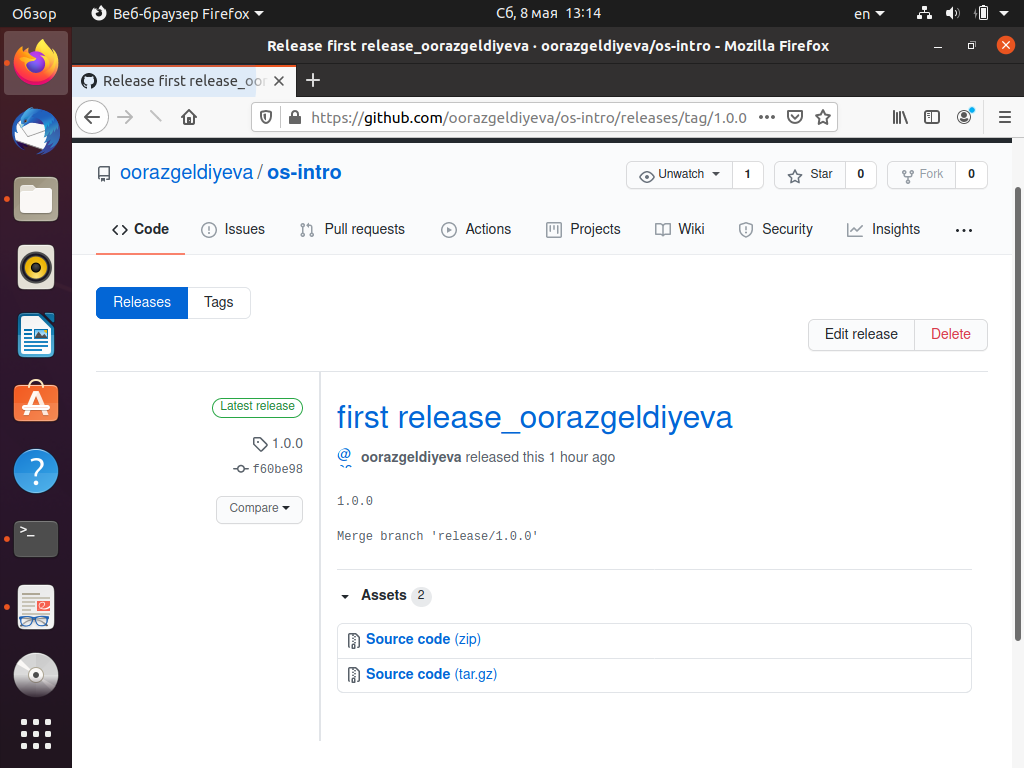


Рисунок 12. Релиз на github

**Вывод:**

на лабораторной работе изучила идеологию и применение средств контроля версий.

**Контрольные вопросы**

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

Ответ:

Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией, позволяющее хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям.

Предназначены для работы нескольких человек над одним проектом, а также при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы.

1. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия

Ответ:

Хранилище – место «памяти», в котором будет храниться новая версия файла после его изменения пользователем.

Commit. В нем содержится описание тех изменений, которые вносит пользователь в код приложения.

История –история изменений. Обычно доступна информация о том, кто из участников, когда и какие изменения вносил.

Рабочая копия – это копия, которую мы выписали в свою рабочую зону, это то, над чем мы работаем в данный момент. Привилегированный доступ только одному пользователю, работающему с файлом

1. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида

Ответ:

Централизованные VCS предполагают наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Пример: AccuRev

Децентрализованные VCS не имеют единого репозитория, он у каждого пользователя свой. Помимо того, они были созданы для обмена изменениями, а не для их объединения. Не имеют какой-то жестко заданной структуры репозиториев с центральным сервером. Пример: Git

1. Опишите действия с VCS при единоличной работе с хранилищем.

Ответ:

При единоличной работе с VCS каждое новое изменение в репозитории сохраняется не со всеми предыдущими версиями. Оно изменяется по системе: одно предыдущее + новая информация

1. Опишите порядок работы с общим хранилищем VCS

Ответ:

1) Создать репозиторий. Это место, где будут лежать файлы. Теперь у нас есть общее хранилище данных, с которым и будет проходить дальнейшая работа.

2) Скачать проект из репозитория

3) Обновить проект, забрать последнюю версию из репозитория

4) Внести изменения в проект

5) Запушить код, т.е изменить код в общем хранилище

6) Создать ветку

1. Каковы основные задачи, решаемые инструментальным средством git?

Ответ:

* Сохранение файлов с исходным кодом
* Защита от случайных исправлений и удалений
* Отмена изменений и удалений, если они оказались некорректными
* Одновременная поддержка рабочей версии и разработка новой
* Возврат к любой версии кода из прошлого
* Просмотр истории изменений

1. Назовите и дайте краткую характеристику командам git.

Ответ:

git init – создание основного дерева репозитория

git push – отправка всех произведённых изменений локального дерева в центральный репозиторий

git status – просмотр списка изменённых файлов в текущей директории

git add – добавить все изменённые и/или созданные файлы и/или каталоги

git add имена\_файлов – добавить конкретные изменённые и/или созданные файлы и/или каталоги

git pull – получение обновлений (изменений) текущего дерева из центрального репозитория

git rm имена\_файлов – удалить файл и/или каталог из индекса репозитория

git commit -am 'Описание коммита' – сохранить все добавленные изменения и все изменённые файлы

git commit – сохранить добавленные изменения с внесением комментария через встроенный редактор

git checkout -b имя\_ветки – создание новой ветки, базирующейся на текущей:

git checkout имя\_ветки – переключение на ветку

git push origin имя\_ветки – отправка изменений конкретной ветки в центральный репозиторий

git merge --no-ff имя\_ветки – слияние ветки с текущим деревом

git branch -d имя\_ветки – удаление локальной уже слитой с основным деревом ветки

git branch -D имя\_ветки – принудительное удаление локальной ветки

git push origin :имя\_ветки – удаление ветки с центрального репозитория

1. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Ответ:

Локальный репозиторий – она же директория “.git”. В ней хранятся коммиты и другие объекты. Локальный репозиторий мы используем, когда работаем одни и нам нужно сохранить свои же изменения

Удаленный репозиторий – тот репозиторий, который считается общим, в который мы можем передать свои коммиты из локального репозитория, чтобы остальные пользователи могли их увидеть. Удаленный репозиторий используется для групповой работы, когда в личном репозитории скопилось достаточно коммитов, мы делимся ими в удаленном для того, чтобы другие пользователи могли видеть наши изменения. Также из удаленного репозитория мы можем скачать чужие изменения

1. Что такое и зачем могут быть нужны ветви (branches)?

Ответ:

Ветка – это подвижный указатель на один из коммитов. Обычно ветка указывает на последний коммит в цепочке коммитов. В ветке мы можем как угодно ломать проект, основной код при этом не пострадает.

1. Как и зачем можно игнорировать некоторые файлы при commit?

Ответ:

Игнорируемые файлы – это, как правило, специфичные для платформы файлы или автоматически созданные файлы из систем сборки.

.gitignore — это простой текстовый файл, в каждой строке которого содержится шаблон, который файлы или каталоги следует игнорировать. Он использует шаблоны подстановки для сопоставления имен файлов с подстановочными знаками