## РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

### Факультет физико-математических и естественных наук

# ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ No 2

Студент: Евсельев Д.  
Группа: НПМбд-01-20  
Преподаватель: Курячиий Г.

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

**Задание**  
Научиться пользоваться системой контроля версий git. Создать репозиторий и опубликовать его на github.

**Выполнение работы**  
Создал учетную запись на github

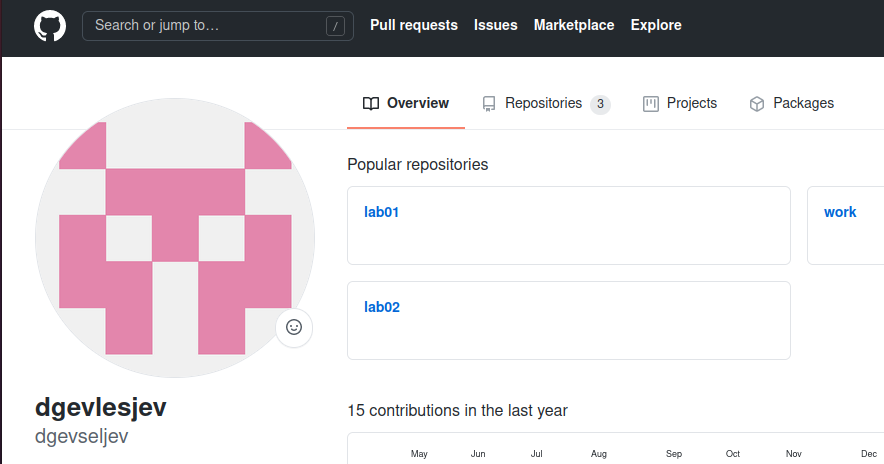


Рис. 1. Учетная запись на github

Рис. 2. Предварительная конфигурация

Рис. 2. Предварительная конфигурация

Далее инициализировал локальный репозиторий, расположенный в tutorial. Создал текстовый файл hello.txt и добавил в репозиторий. Воспользовался командой git status для просмотра изменений.

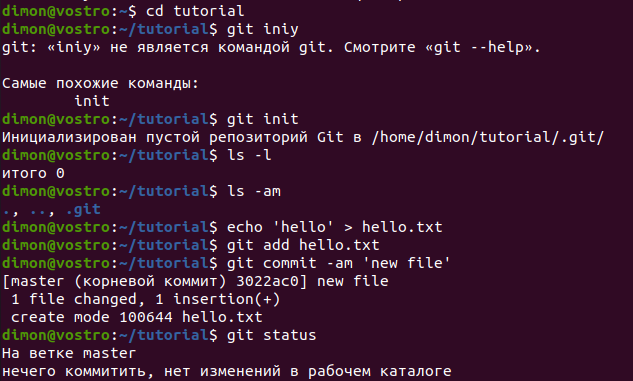


Рис. 3. Добавление файла в локальный репозиторий

Прописал шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore

Рис. 4. Установка шаблонов игнорируемых типов файлов

Рис. 4. Установка шаблонов игнорируемых типов файлов

Сгенерировал ключ для идентификации пользователя на сервере репозитариев.

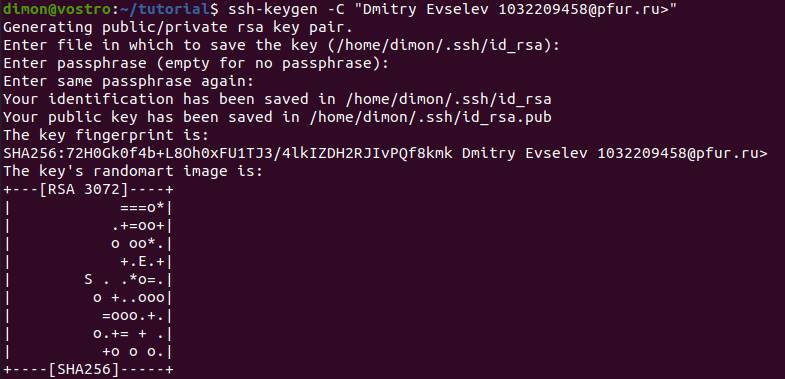


Рис. 5. Генерирование ключа.

Указал ключ в github

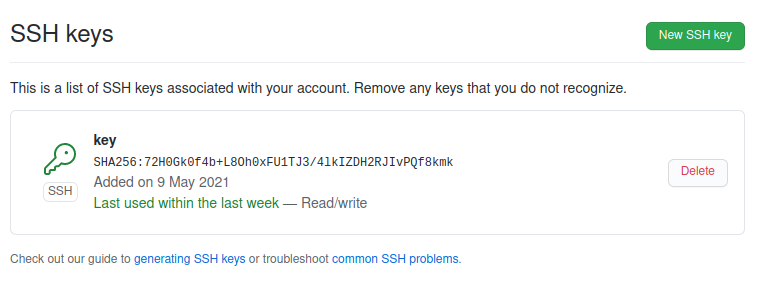


Рис. 6. Указание ключа на github

Загрузил репозиторий из локального каталога на сервер

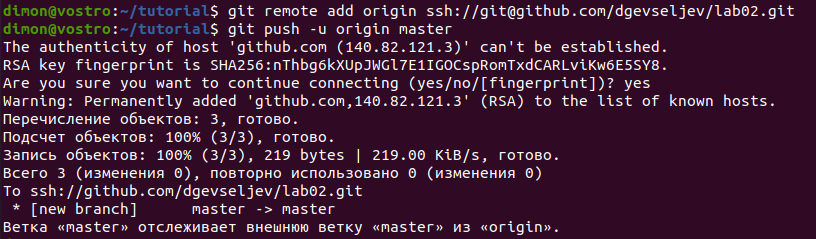


Рис. 7. Загрузка локального репозитария на сервер

Создал заготовку для файла README.md, закоммитил и выложил на github

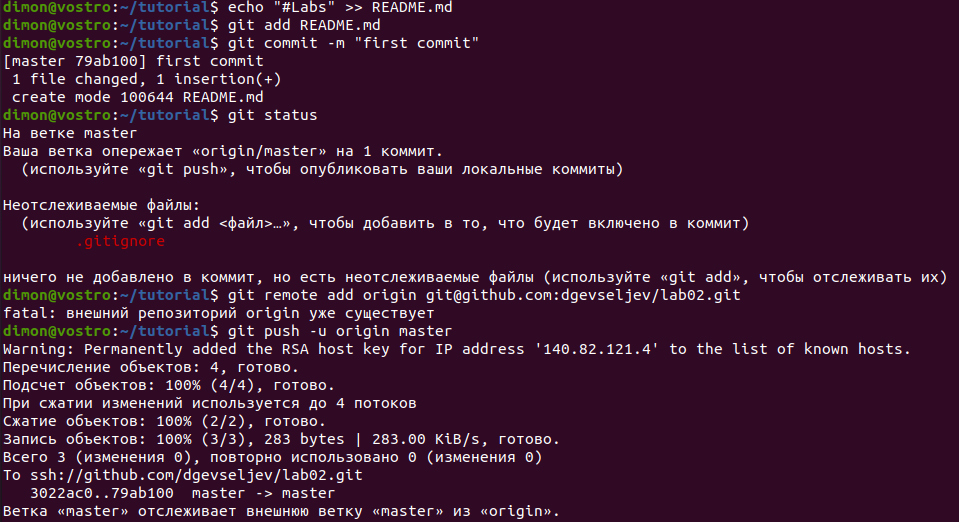


Рис. 8. Добавление README.md в репозиторий на нитхабе

Далее добавил файл лицензии, шаблон игнорируемых файлов, закоммитил и отправил на github



Рис. 9.1. Скачивание лицензии и шаблонов игнорируемых файлов

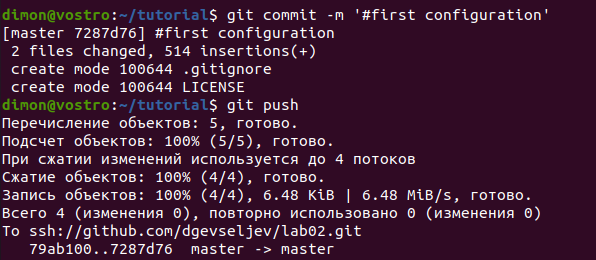


Рис. 9.2. Коммит и отправка на github

Далее идет работа с git-flow

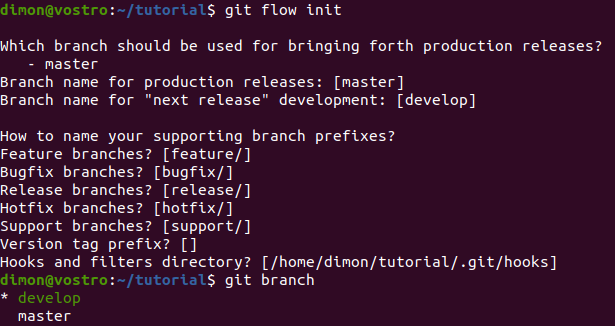


Рис. 10. Инициализация git-flow

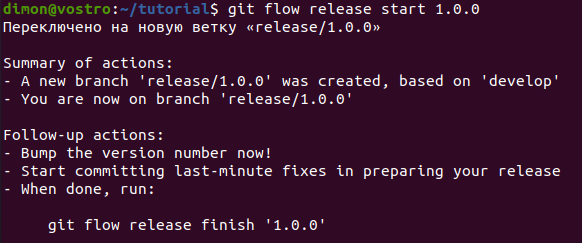


Рис. 11. Старт релиза версии 1.0.0

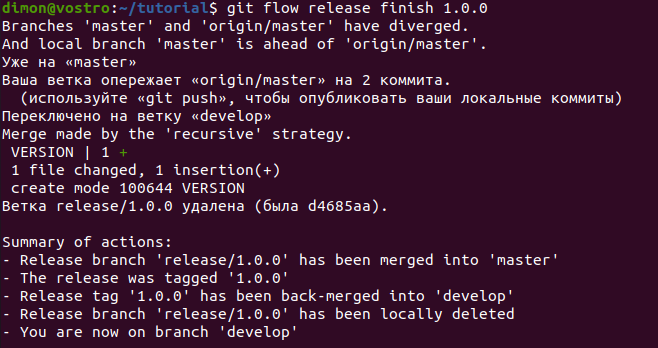


Рис. 12. Финиш релиза версии 1.0.0

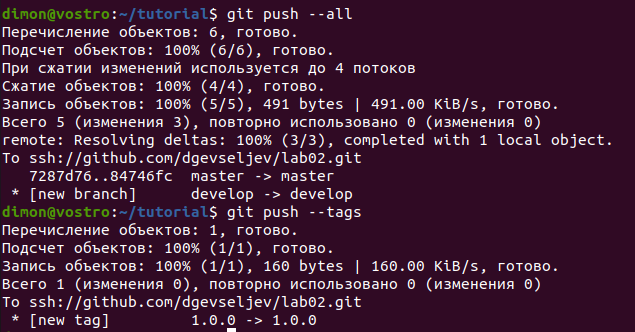


Рис. 13. Отправка данных на github

**Вывод**  
Научился пользоваться системой контроля версий git. Создал репозиторий и опубликовал его на github.

**Ответы на контрольные вопросы**  
1. **Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?**  
Программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение.  
2. **Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.**  
Хранилище – место где хранятся файлы после внесения изменений,  
commit – сохранение добавленных изменений,  
история – история изменений файлов,  
рабочая копия – копия, которая находится в работе.  
3. **Что представляют собой и чем отличаются централизованные и децентрализованные VCS?**  
Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение.  
Децентрализованные системы контроля версий позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой.  
4. **Опишите действия с VCS при единоличной работе с хранилищем.**  
Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельтакомпрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.  
5. **Опишите порядок работы с общим хранилищем VCS.**  
Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручнуювыбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.  
6. **Каковы основные задачи, решаемые инструментальным средством git?**  
Git даёт возможность разработчикам отслеживать изменения в файлах и работать над одним проектом совместно с коллегами.  
7. **Назовите и дайте краткую характеристику командам git.**  
– создание основного дерева репозитория: git init  
– получение обновлений (изменений) текущего дерева из центрального репозитория: git pull  
– отправка всех произведённых изменений локального дерева в центральный репозиторий: git push  
– просмотр списка изменённых файлов в текущей директории: git status  
– просмотр текущих изменения: git diff  
– добавить все изменённые и/или созданные файлы и/или каталоги: git add .  
– добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add имена\_файлов  
– удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена\_файлов  
– сохранить все добавленные изменения и все изменённые файлы: git commit -am ‘Описание коммита’  
– сохранить добавленные изменения с внесением комментария через встроенный редактор: git commit  
– создание новой ветки, базирующейся на текущей: git checkout -b имя\_ветки  
– переключение на некоторую ветку: git checkout имя\_ветки (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)  
– отправка изменений конкретной ветки в центральный репозиторий: git push origin имя\_ветки  
– удаление локальной уже слитой с основным деревом ветки: git branch -d имя\_ветки  
– принудительное удаление локальной ветки: git branch -D имя\_ветки  
– удаление ветки с центрального репозитория: git push origin имя\_ветки  
8. **Приведите примеры использования при работе с локальным и удалённым репозиториями.**  
Если я забыл, в какой ветке нахожусь, то с помощью git branch могу посмотреть это. Если мне нужно подключить систему контроля версий к уже существующему проекту, то я инициализирую локальный репозиторий gitinit и подключаю удаленный git remote add, затем добавляю все файлы git add и коммичу их git commit, затем пушу на удаленный репозиторий git push. Теперь к моему проекту подключена система контроля версий  
9. **Что такое и зачем могут быть нужны ветви (branches)?**  
Ветки нужны для разделения разработки. Например, когда разрабатывается новая фича, не нужно, чтобы она присутствовала в основном проекте, поэтому для нее создают отдельную ветку. В случае успешной разработки фичи, эту ветку сливают с основной. Так убираются риски багов, ошибок, а также утечки данных  
10. **Как и зачем можно игнорировать некоторые файлы при commit?**  
Есть временные и системные файлы, которые засоряют проект и не нужны. путь к ним можно добавить в файл .gitignore, тогда они не будут добавляться в проект.