# Отчёт о выполнении лабораторной работы №2 Управление версиями

***Российский Университет Дружбы Народов***  
***Факульткт Физико-Математических и Естественных Наук***

***Дисциплина:*** *Операционные системы*

***Работу выполнял:*** *Адебайо Ридван* *1032205020*

*НКНбд-01-20*

***Москва. Дисплейный класс РУДН. 2021г.***

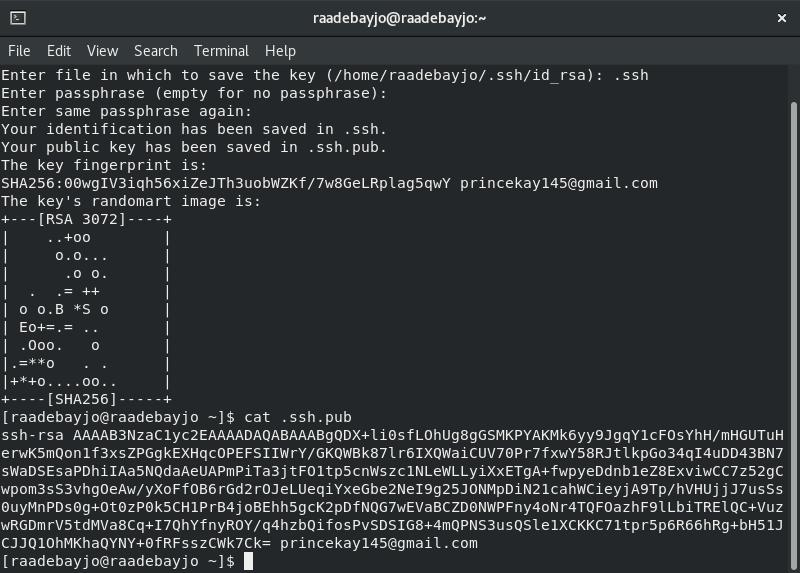
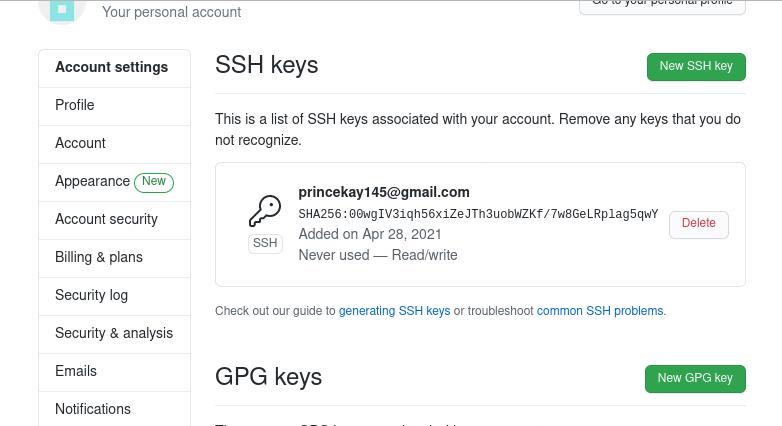
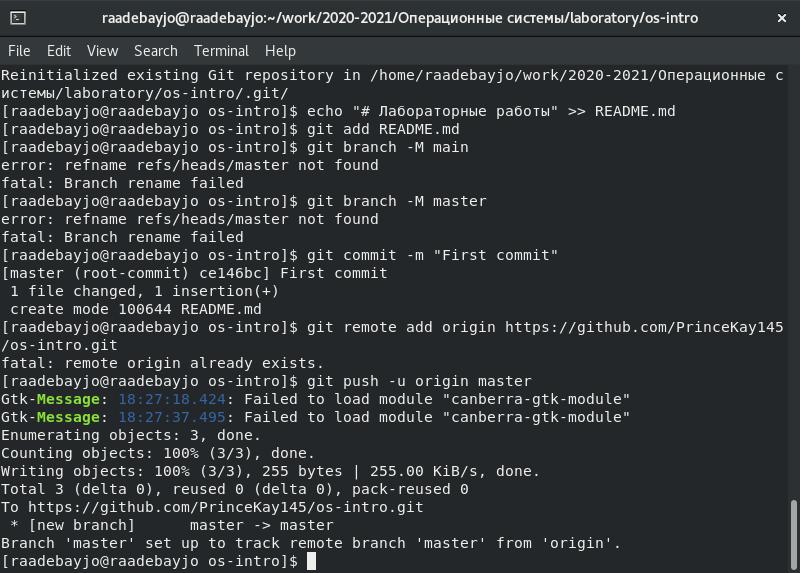
## Цель работы

Изучить идеологию и применение средств контроля версий. ## Задание 1. Создайте учётную запись на https://github.com.  
2. Настройте систему контроля версий git, как это описано выше c использованием сервера репозиториев https://github.com/.  
3. Создайте структуру каталога лабораторных работ согласно пункту М.2.  
4. Подключение репозитория к github  
– Создайте репозиторий на GitHub. Для примера назовём его os-intro.  
– Рабочий каталог будем обозначать как laboratory. Вначале нужно перейти в этот каталог:  
– Инициализируем системы git:  
– Создаём заготовку для файла README.md:  
echo “# Лабораторные работы” >> README.md  
git add README.md

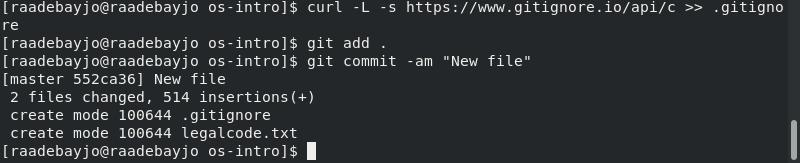
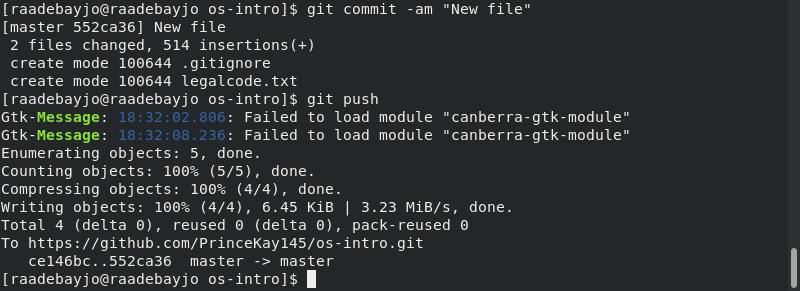
– Делаем первый коммит и выкладываем на github:  
git commit -m “first commit”  
git remote add origin  
↪ git@github.com:/sciproc-intro.git  
5. Первичная конфигурация  
– Добавим файл лицензии: wget https://creativecommons.org/licenses/by/4.0/legalcode.txt  
– Добавим шаблон игнорируемых файлов. Просмотрим список имеющихся шаблонов:  
curl -L -s https://www.gitignore.io/api/list  
Затем скачаем шаблон, например, для C:  
curl -L -s https://www.gitignore.io/api/c >> .gitignore  
Можно это же сделать через web-интерфейс на сайте https://www.gitignore.  
io/. – Добавим новые файлы:  
git add .  
– Выполним коммит:  
git commit -a  
– Отправим на github:  
git push  
6. Конфигурация git-flow  
– Инициализируем git-flow  
git flow init  
Префикс для ярлыков установим в v.  
– Проверьте, 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  
git push –tags  
– Создадим релиз на github.

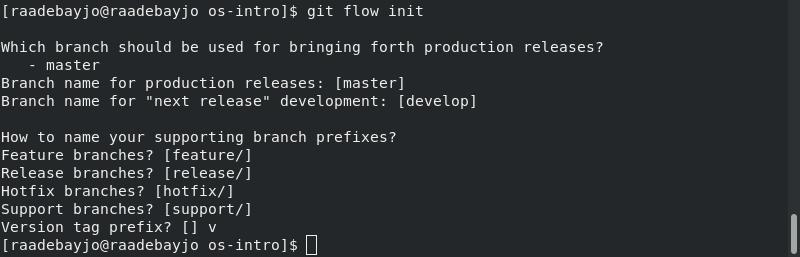
## Выполнение работы

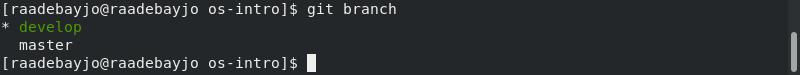
1. Создала учётную запись на https://github.com.
2. Настройте систему контроля версий git добавила ssh ключ

* сгенерировал пару ключей 
* Добавил SSH-key  
  
* Cделал первый коммит 

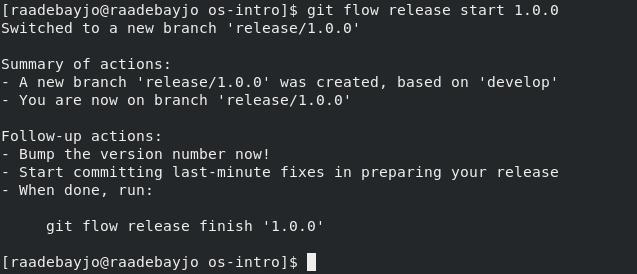
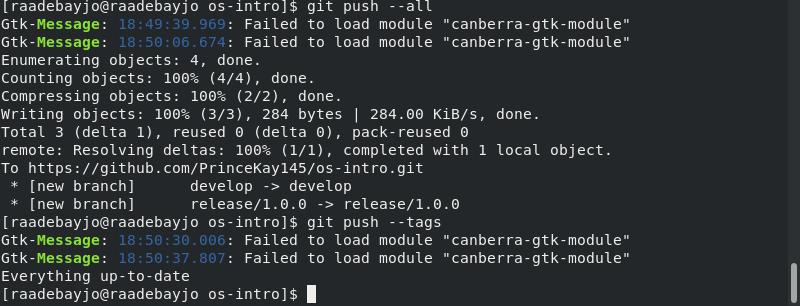
1. Первичная конфигурация

* Добавил новый файл и коммит 
* Пуш на GitHub 

1. Конфигурация git-flow  
   -Инициализировал git-flow 

* Проверил, что я на ветке develop 

-Создал релиз с версией 1.0.0:

 - Записал версию Добавил в индекс:  
Залил релизную ветку в основную ветку  
Отправил данные на github  
Создадим релиз на github. 

# Выводы

Выполняя данную лабораторную работу, я научился пользоваться Git, а именно: инициализировать локальный репозиторий, подключать удаленные репозитории, добавлять и удалять нужные файлы, синхронизировать данные. Также я научился использовать Git Flow, который очень сильно упрощает разработку проектов и навигацию между ветками

# Контрольные вопросы

1. Системы контроля версий - это программное обеспечение, которое используется для облегчения работы с изменяющейся информацией, обычно - с проектами. Чаще всего применяется в разработке, когда над одним проектом работает большое количество людей
   * Хранилище в системе контроля версий - это удаленный репозиторий, где хранятся все файлы проекта
   * commit - это фиксация изменений перед загрузкой файлов в систему контроля версий
   * история хранит все изменения проекта, и в случае необходимости позволяет откатиться к нужному месту
   * рабочая копия - это копия проекта на компьютере разработчика. Если другой член команды изменил проект, необходимо загрузить новую версию проекта себе на компьютер
2. Централизованные системы контроля версий хранят данные о проекте на едином сервере, и в случае его отключения, доступ к данным будет утерян (Perforce) В децентрализовынных системах у каждого из участников проекта на компьютере хранитсяя полная копия проекта, что позволяет меньше зависеть от сервера (Git)
3. Сначала надо создать и подключить удаленный репозиторий. Затем, т.к. никто кроме тебя не изменяет проект, по мере изменения проекта пушить изменения на сервер, и нет необходимости загружать изменения
4. Каждый раз перед разработкой необходимо загрузить актуальную версию проекта на свой компьютер, а уже потом раюотать над ним. После работы необходимо закоммитить изменения и запушить на сервер
5. Упрощение обмена информацией, ускорение разработки, устранение ошибок и недоработок во время разработки.
   * git init - инициализирует локальный репозиторий
   * git add - добавляет файлы в репозиторий
   * git commit - коммит версии
   * git pull - загружает актуальную версию проекта
   * git push - отправляет измененный проект на сервер
   * git checkout - позволяет переключаться между ветками
   * git status - текущий статус проекта
   * git branch - просмотр доступных веток
   * git remote add - добавление удаленного репозитория
6. 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 git init 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
7. 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
8. 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