**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №**  **2**

*дисциплина:* *Операционные системы*

Студент: Артамонов Т.Е.

Группа: НКНбд-01-21

**МОСКВА**

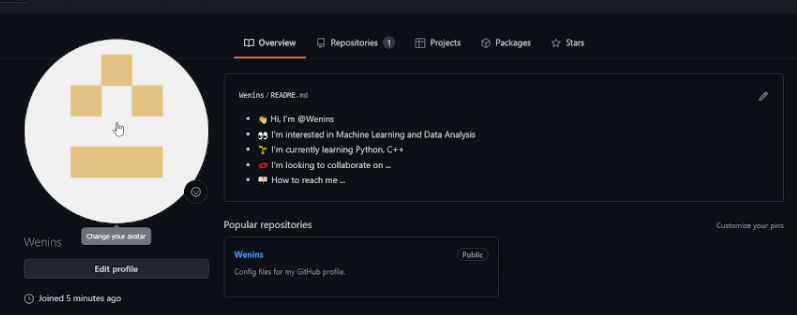
2022 г.

Отчет по выполнению лабораторной работы №2

**Тема:** Управление версиями

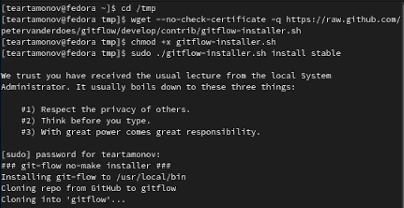
**Цель работы:** Изучение идеологии и применение средств контроля версий. Освоение умений по работе с git.  
**Выполнение работы:**

**Настройка github**

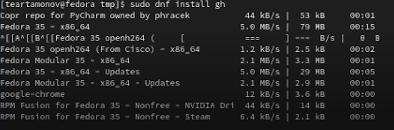


Регистрируем учётную запись github и заполняем основные данные.

**Установка программного обеспечения**



Устанавливаем git-flow в Fedora.



Устанавливаем gh.

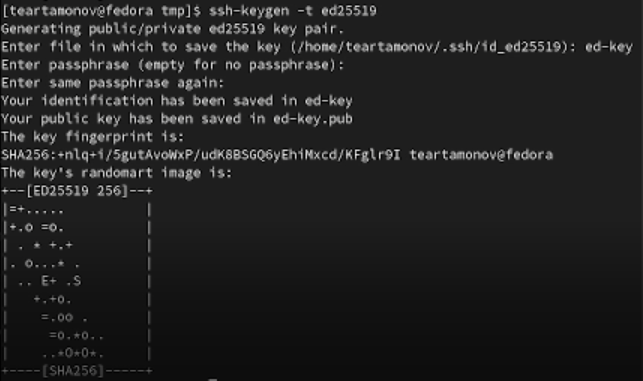
**Настройка git**



Зададим имя и email владельца репозитория, то есть нас. 

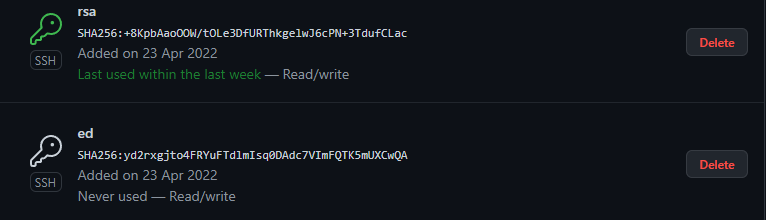
Зададим имя начальной ветки и параметры.

**Создание ключей ssh и добавление их в GitHub**

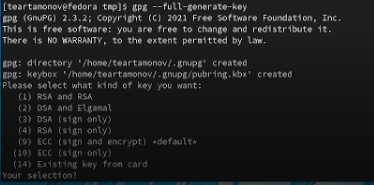


Создаем ed25519 ключ и точно так же rsa. 

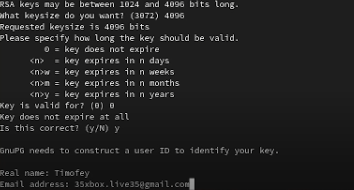
Копируем ключи в буфер и по очереди добавляем их в гитхаб.



**Создание и добавление GPG ключа в GitHub**

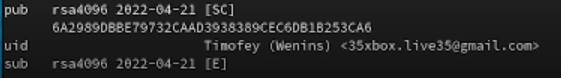


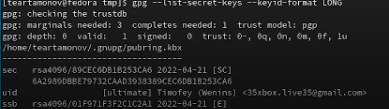
Создаём ключ и настраиваем его, так, как сказано в руководстве лабораторной работы.



Указываем такой же адрес, как и на гитхабе.



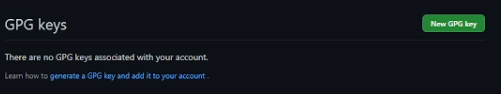




Получаем отпечаток нашего ключа.

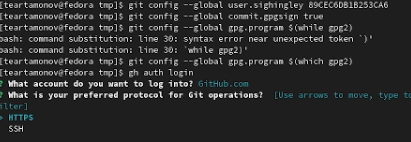


Копируем его в буфер.



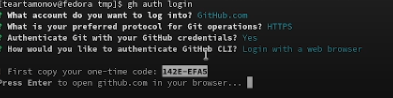
Добавляем его в гитхаб.

**Настройка автоматических подписей коммитов git**



Настраиваем автоматические подписи с помощью нашей почты.

**Настройка gh**



Входим в гитхаб через гит.

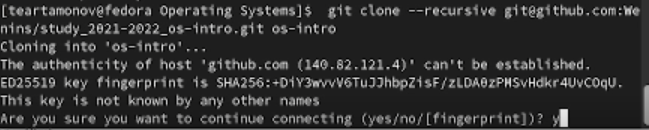
**Шаблон для рабочего пространства**



Создаем папку.

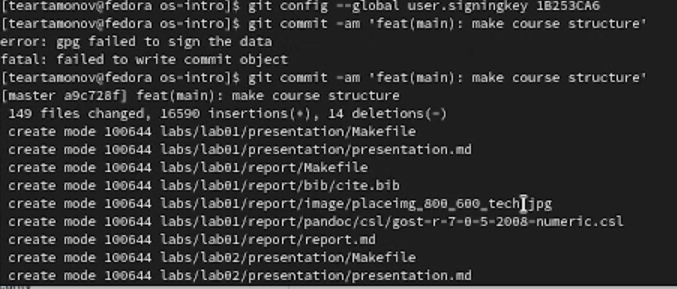


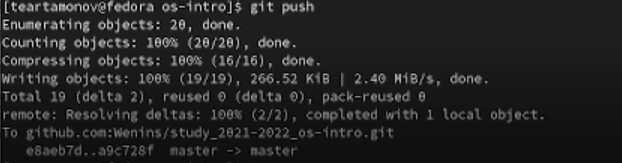
Копируем шаблон.





Удаляем лишние файлы и создаем необходимые папки.





Отправляем файлы на сервер.

**Контрольные вопросы**

1. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.
2. Хранилище - место в репозитории, где размещаюся все версии проекта и прочее.  
   Под commit понимается новая версия/изменения проекта.  
   История - список всех версий.  
   Рабочая копия - версия проекта, на которой в данный момент ведется работа.  
   Все эти явления связаны между собой и вместе представляют собой git.
3. Централизованная vcs – vcs с единым репозиторием. Примером может выступать репозиторий отдела баланса в видеоиграх.  
   Децентрализованная vcs – vcs с несколькими репозиториями. Пример - если участники не работают над одним проектом, например учебная группа в ВУЗе.
4. Человек просто работает с файлом и создает для него различные версии, возвращаясь после к лучшей из конкурентов и продолжает работать уже над ней.
5. С общим хранилищем, участники могут предлагать свои идеи по усовершенствованию файла, удачные добавляются и становятся частью нового файла.
6. Такие же как и у всех VCS.
7. Создание основной ветки git init   
   получение обновлений 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 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   
   слияние ветки с текущим деревом git merge --no-ff 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. git add .  
   git rm
9. Ветки нужны для возможного отката версии или же просто для будущего выбора между ними нового ядра проекта.
10. Некоторые файлы это файлы, генерируемые машиной для работы и как правило они не принимают прямого участия в разработке.