Лабораторная работа 2

Управление версиями

Лебедева Ольга Андреевна

Содержание

# 1 Цель работы

– Изучить идеологию и применение средств контроля версий.

– Освоить умения по работе с git.

# 2 Указания к лабораторной работе

Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями.

# 3 Последовательность выполнения работы

Настройка github

1. Создайте учётную запись на https://github.com.
2. Заполните основные данные на https://github.com. (рис. 1)

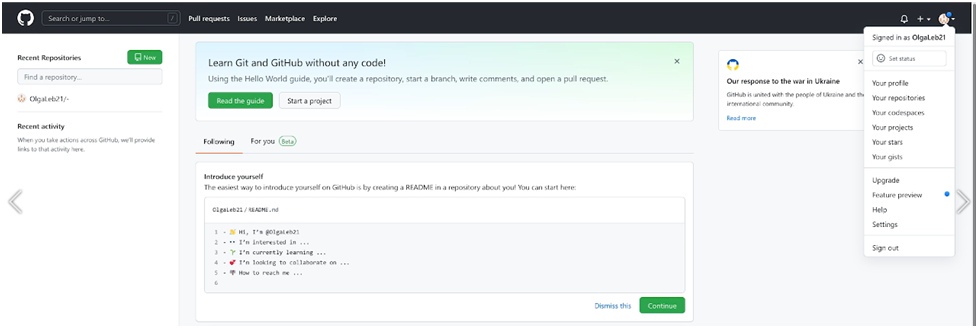


Рис. 1: Создание профиля на Гитхаб

# 4 Базовая настройка git

– Зададим имя и email владельца репозитория: (рис. 2)

git config –global user.name “Name Surname”

git config –global user.email “work@mail”

Рис. 2: Имя и email репозитория

Рис. 2: Имя и email репозитория

Настроим верификацию и подписание коммитов git.

– Зададим имя начальной ветки (будем называть её master):(рис. 3)

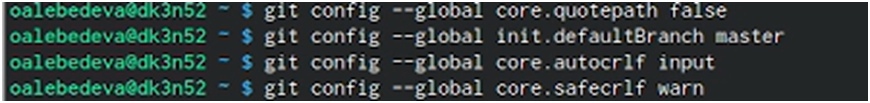


Рис. 3: Задаём имя ветки и параметры

Создание ключей ssh

Генерируем ключ (рис. 4)

gpg –full-generate-key

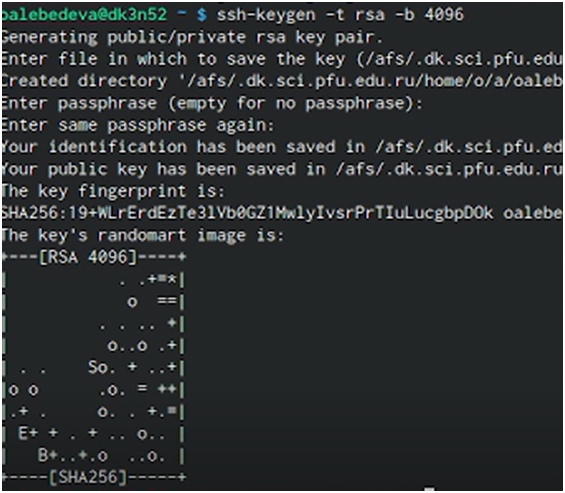


Рис. 4: ключ ssh

Из предложенных опций выбираем:(рис. 5) – тип RSA and RSA;

– размер 4096;

– выберите срок действия; значение по умолчанию— 0 (срок действия не истекает никогда).

– GPG запросит личную информацию, которая сохранится в ключе: – Имя (не менее 5 символов).

– Адрес электронной почты.

– При вводе email убедитесь, что он соответствует адресу, используемому на GitHub.

– Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

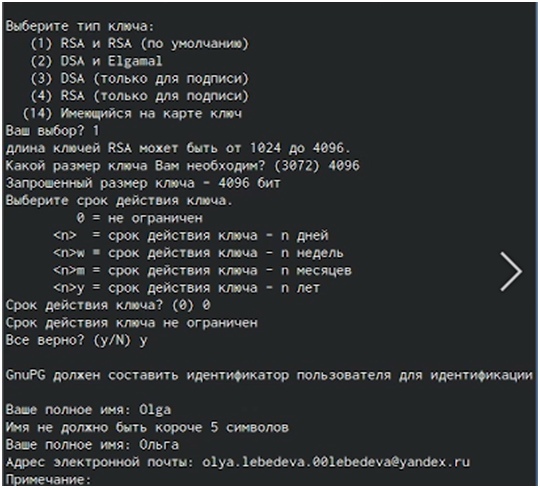


Рис. 5: Опции ключа

# 5 Добавление PGP ключа в GitHub

– Выводим список ключей и копируем отпечаток приватного ключа:(рис. 6)

gpg –list-secret-keys –keyid-format LONG

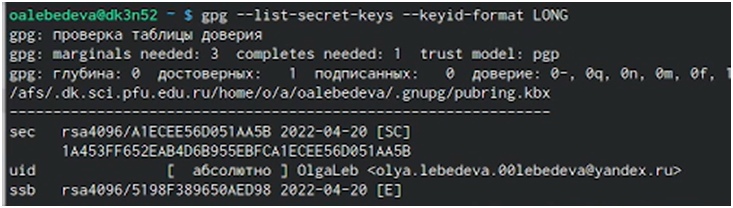


Рис. 6: Вывод ключа

Перейдём в настройки GitHub (https://github.com/settings/keys), нажмём на кнопку New GPG key и вставим полученный ключ в поле ввода.(рис. 7)

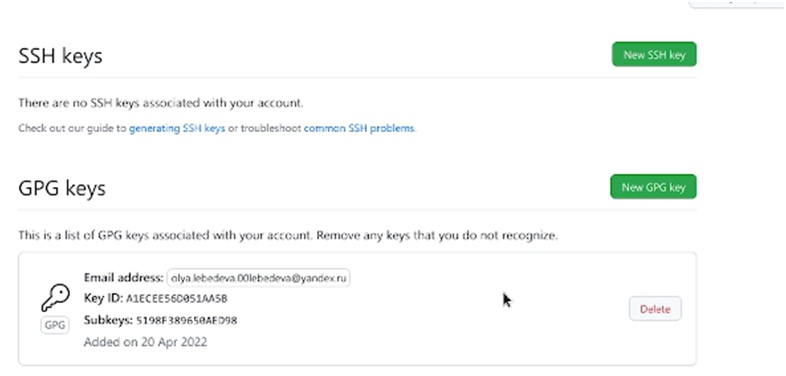


Рис. 7: Добавление ключа GPG

# 6 Создание репозитория

Шаблон для рабочего пространства – Репозиторий: https://github.com/yamadharma/course-directory-student-template.(рис. 8)

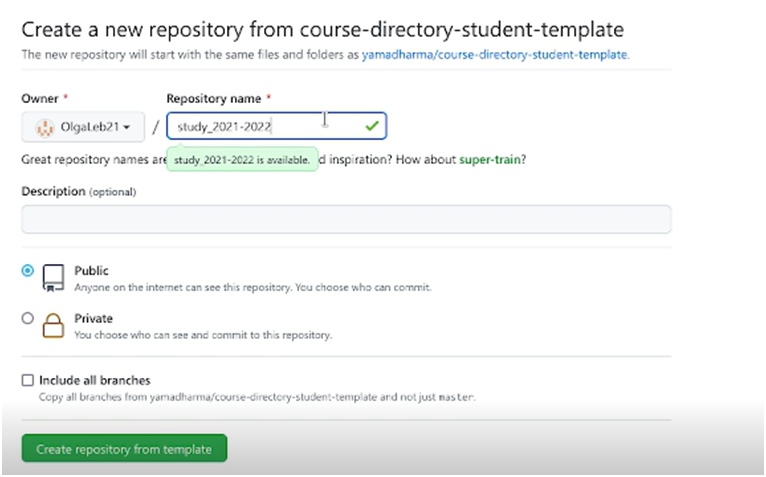


Рис. 8: Создание репозитория

Создание репозитория примет следующий вид:

mkdir -p ~/work/study/2021-2022/“Операционные системы”

cd ~/work/study/2021-2022/“Операционные системы”

gh repo create study\_2021-2022\_os-intro –template=yamadharma/course-directory-student-template –public

git clone –recursive git@github.com:/study\_2021-2022\_os-intro.git os-intro

Команда gh оказалась недоступной для использования, поэтому мы пошли другим путём:(рис. 9)

Клонируем репозиторий, представленный в виде шаблона.(рис. 11)

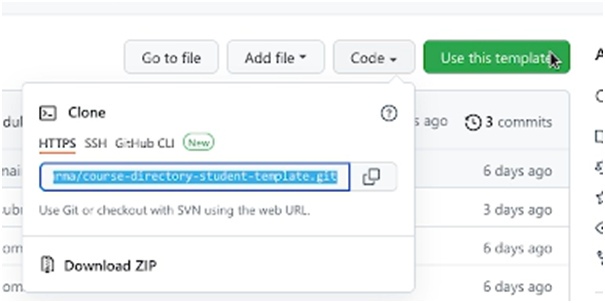


Рис. 9: Копируем ссылку на репозиторий

Далее, аналогично ключам GPG впишем ключи SSH:(рис. 10)

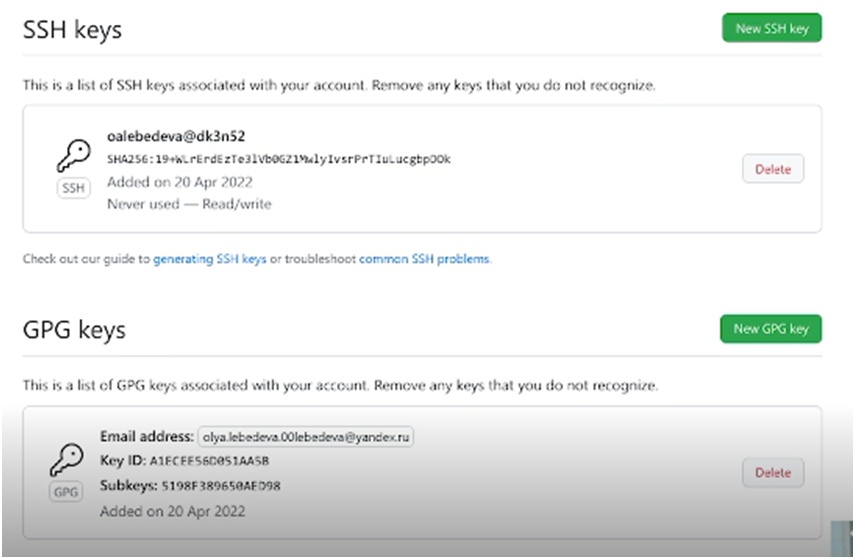


Рис. 10: Добавление первых ключей

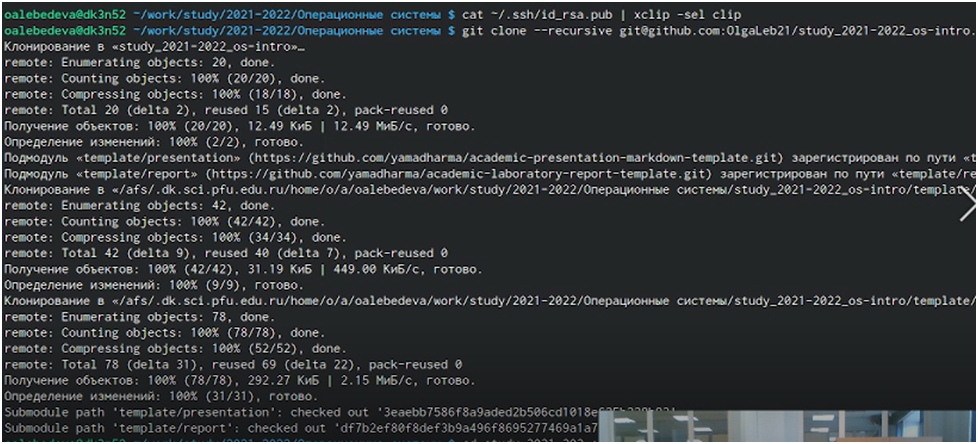


Рис. 11: Клонирование файлов в репозитория

# 7 Настройка каталога курса

Перейдем в каталог курса и удалим лишние файлы:(рис. 12)

Рис. 12: Удаление файла

Рис. 12: Удаление файла

Отправим файлы на сервер:(рис. 13)

git add .

git commit -am ‘feat(main): make course structure’

git push

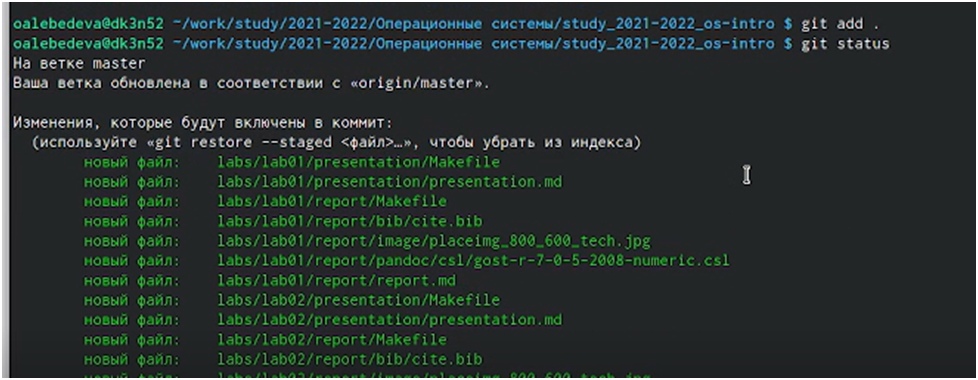


Рис. 13: Обновление ветки (из-за возникновения ошибки)

# 8 Настройка автоматических подписей коммитов git

– Используя введёный email, укажем Git применять его при подписи коммитов:(рис. 14) (рис. 15)

git config –global user.signingkey

git config –global commit.gpgsign true

git config –global gpg.program $(which gpg2)

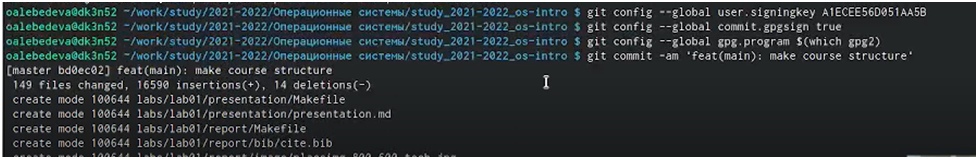


Рис. 14: Нахождение новых файлов в папках

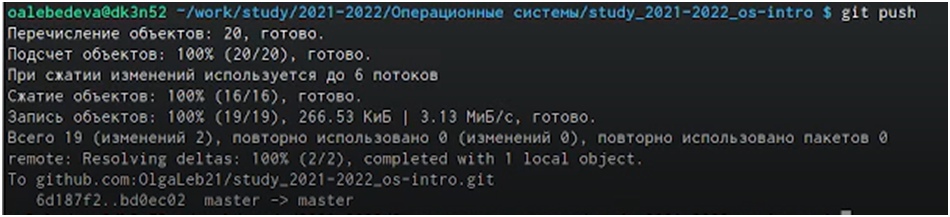


Рис. 15: Отправка файлов в репозиторий

# 9 Вывод

– Изучили идеологию и применение средств контроля версий.

– Освоили умения по работе с git.

# 10 Контрольные вопросы

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

Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии. Для контроля версий файлов в этой книге в качестве примера будет использоваться исходный код программного обеспечения, хотя на самом деле вы можете использовать контроль версий практически для любых типов файлов.

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

Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией Commit («[трудовой] вклад», не переводится) — процесс создания новой версии Рабочая копия (working copy) — текущее состояние файлов проекта, основанное на версии, загруженной из хранилища (обычно на последней). Версия (revision), или ревизия, — состояние всех файлов на определенный момент времени, сохраненное в репозитарии, с дополнительной информацией

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

Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. (Пример — Wikipedia.) В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. (Пример — Bitcoin)

1. Опишите действия с VCS при единоличной работе с хранилищем.(рис. 16)

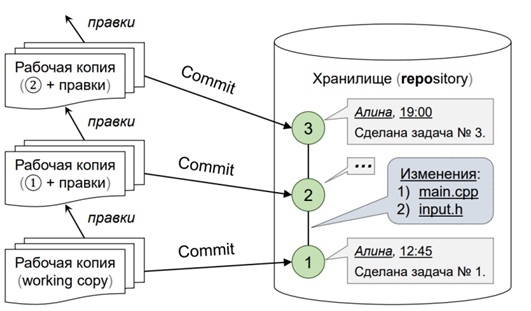


Рис. 16: Eдиноличная работа с хранилищем

1. Опишите порядок работы с общим хранилищем VCS. (рис. 17)

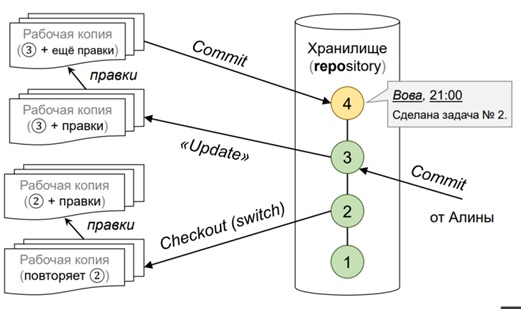


Рис. 17: Работы с общим хранилищем

1. Каковы основные задачи, решаемые инструментальным средством git?

У Git есть две основные задачи: хранить информацию обо всех изменениях в коде, начиная с самой первой строчки, и обеспечить удобства командной работы над кодом.

1. 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

– сохранение текущих изменений: – 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

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

Локальный репозиторий — она же директория “.git”. В ней хранятся коммиты и другие объекты. Удаленный репозиторий – тот самый репозиторий который считается общим, в который вы можете передать свои коммиты из локального репозитория, что бы остальные программисты могли их увидеть. Удаленных репозиториев может быть несколько, но обычно он бывает один.

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

‘Git branch’ – это команда для управления ветками в репозитории Git. Ветка – это просто «скользящий» указатель на один из коммитов. Когда мы создаём новые коммиты, указатель ветки автоматически сдвигается вперёд, к вновь созданному коммиту. Ветки используются для разработки одной части функционала изолированно от других. Каждая ветка представляет собой отдельную копию кода проекта. Ветки позволяют одновременно работать над разными версиями проекта. Ветвление («ветка», branch) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). Ветки нужны для того, чтобы программисты могли вести совместную работу над проектом и не мешать друг другу при этом.

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

Игнорируемые файлы обычно представляют собой файлы, специфичные для платформы, или автоматически созданные из сборочных систем. Временно игнорировать изменения в файле можно командой: git update-index —assume-unchanged